ENgineering and TEchnology institute Groningen

Analysis and comparison of wave energy extraction in the Ocean Grazer's wave tank experimental setup

Master's Thesis

Genís Martí Manresa

First supervisor: Bayu Jayawardhana
Second supervisor: Antonis Vakis

Industrial Engineering

Groningen, June 2017

Abstract

Fossil fuels have been fundamental to the evolution of humanity's technology and way of life for the last two centuries. However, their use has two main disadvantages: (a) the climate change produced by the release of carbon dioxide (CO_2) during their combustion; and (b) their limited reserves. In the last decades, society has tried to counter these problems by potentiating the use of renewable energies instead of fossil fuels.

The Ocean Grazer (OG) is one of the many proposed projects to develop the renewable energy field further. Developed by the University of Groningen, the OG platform is expected to extract and store multiple forms of renewable energy, of which wave energy is its primary source. The main innovation provided by the OG is a novel Wave Energy Converter (WEC) technology denominated MP^2PTO, which aims to adapt to the different wave profiles so the maximum energy content can always be extracted.

In the last years, one of the many research lines in the group has been the study of the wave energy extraction process in an experimental setup called the *wave tank*, which contains a prototype of the multi-piston PTO concept used in the OG-WEC. More precisely, in the last few months, experimental measurements of the velocity of the particles within the tank have been carried out using the Digital Particle Image Velocimetry (DPIV) technique, taking into account different scenarios that allow investigating the influence of the prototype in the process of energy extraction.

In this work, the validation of these experimental measurements of the kinetic energy contained in the OG group's wave tank experimental setup is approached by the use of a computational fluid dynamics (CFD) simulation software developed by the University of Groningen called ComFLOW.

In a first phase of the validation process, a time domain analysis of the simulation results is done. In this analysis, the non-convergence of the simulation is detected. However, the results of the simulations are compared to the experimental measurements leading to a qualitative coincidence of the evolution of the content of energy along the tank but not quantitative.

In a second phase, the non-convergence of the simulation is studied by means of a frequency domain analysis of the wave tank's water surface height. This study allows the detection of a strong mesh-dependant wave reflection in the simulation's wave tank, which is the reason behind the non-convergence of the simulation.

Consequently, due to this non-convergence, no validation of the experimental measurements can be fully confirmed even though the behaviour of the kinetic energy along the simulation tank is very similar to the experimental measurements. Finally, further research paths are proposed in the Conclusion Section.

Acknowledgements

First, I would like to thank prof. dr. Bayu Jayawardhana as my first supervisor for opening for me the gates of the Ocean Grazer project and guiding me in the effort of producing this thesis.

On second place, I would like to express my gratitude to Henk Seubers and Yianji Wei. In the first case, for helping me especially in the first months of my stay at the University of Groningen with the matters related to the ComFLOW software. In the second case, for his incalculable help in the frequency domain analysis of the results and on how to develop the code used to generate the wavelet transform and harmonic analysis plots.

Next, I would like to thank the whole Ocean Grazer team, for giving their insight on the evolution of my work and helping me find the way to finish it. Particularly to Jesus Barradas for his mentoring on the firsts days of my stay as well as with the writing of the thesis, Marijn van Rooij for being always accessible when I needed information about the wave tank experimental setup and Antonis Vakis as my second supervisor for always giving good advice in the group's meetings.

I would like to thank also all the people with whom I have shared my time with during my stay in Groningen: Álvaro, Matthieu, Eeli, Simon, Petra... And, especially, to Adrián, with whom I have shared student house and the University office, for all the discussions we've had about the Ocean Grazer and other topics.

Finally, I would like to thank my parents, Josep and Concepcio, for giving me always all their support; and Raquel, for being always there.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fossil fuels (Oil, Natural gas, and Coal) have been fundamental to the evolution of humanity's technology and way of life for the last two centuries. Nowadays, this kind of resources still corresponds to an important share of the total energy consumed in the world.

However, the heavy consumption of fossil fuels, and the associated release of carbon dioxide (CO_2) and other greenhouse gases into the atmosphere, alter the chemistry of said atmosphere at an unprecedented rate. The former changes the natural cycle of global atmospheric molecules and accelerates the overall warming trend on the planet at a rate never seen before [1].

This climate change affects all the levels of biodiversity and has effects on individuals, populations, species, ecological networks, and ecosystems. At the most basic levels, "climate change is able to decrease genetic diversity of populations due to directional selection and rapid migration, which could, in turn, affect ecosystem functioning and resilience" [2].

On the other hand, fossil fuels reserves are limited. Their formation began hundreds of millions of years ago with the burial of plants and other living organisms of the period. Once buried, said organisms started a slow decomposition process that, with the passing of time, transformed them into the fossil fuel available today. The extraction rate of said reserves has been, since the beginning of the industrial revolution, way higher than the rate in which nature is able to produce the same amount of fuel. That means that in a few years from now there will not be any fossil fuel left to extract.

In fact, taking into account the current production of each kind of fuel and an approximation of the reserves still existent, an estimation of the amount of years that we will be able to extract fuel still can be made. Nowadays there is only enough oil, natural gas and coal reserves as to meet 50, 52 and 153 years of current production, respectively [3]. Nevertheless, these approximations do not have a linear evolution as it could be expected, as shown in Figure 1.1. This is due to (a) the fact that energetic consumption is growing year after year, (b) the discovery of new reservoirs or (c) the apparition of new extraction technologies that make possible to extract more fuel from the existent resources.

For the last few decades, society has tried to counter these two problems attached with fossil fuels. One of the approaches to deal with them is by potentiating the use of renewable energies on top of fossil fuels.

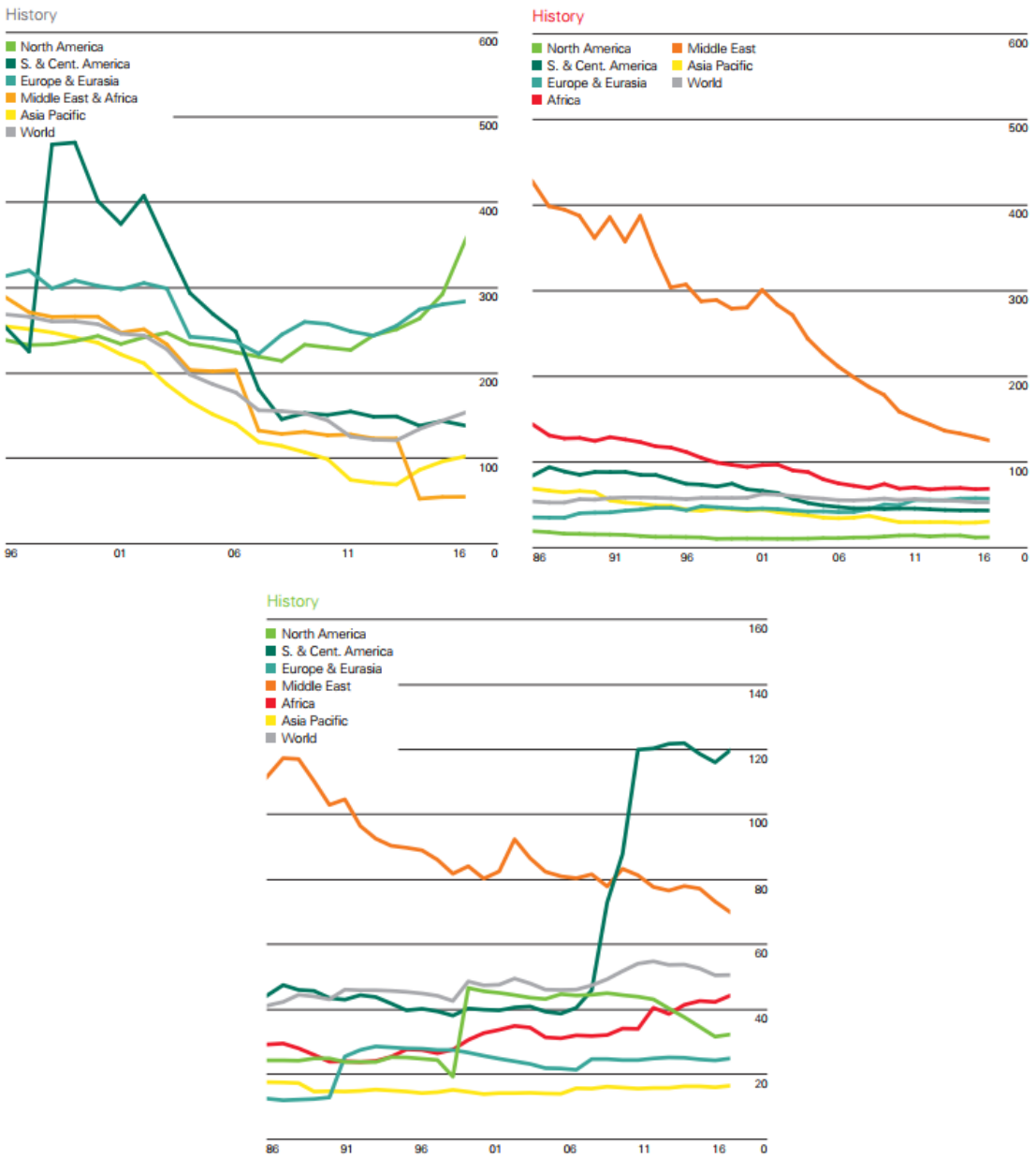

Figure 1.1: Historical evolution, from 1986 to 2016, of the prediction of remaining years of fuel production for Coal (Top left), Natural Gas (Top right) and Oil (Bottom). [3]

1.1 Renewable energy

Renewable energies are energy sources that are continually replenished by nature at a similar rate as they are consumed [4]. In consequence, by definition, renewable energy sources are not depleted when used.

In the majority of types of renewable energy, the original energy source is the same, the sun. It is possible to produce energy that derives directly from it, like in the case of thermal, photochemical, and photo-electric energy; indirectly, which is the case of the wind, hydropower and photosynthetic energy stored in biomass; or from other natural movements and mechanisms of the environment, such as geothermal and ocean energy.

The potential of these energy sources is enormous as it can meet many times the world's demand. A demand which, as it is shown in Figure 1.2, is going to increase considerably in the coming years (primarily due to population growth). Consequently, renewable-based energy systems will be of great help to meet the projected estimations.

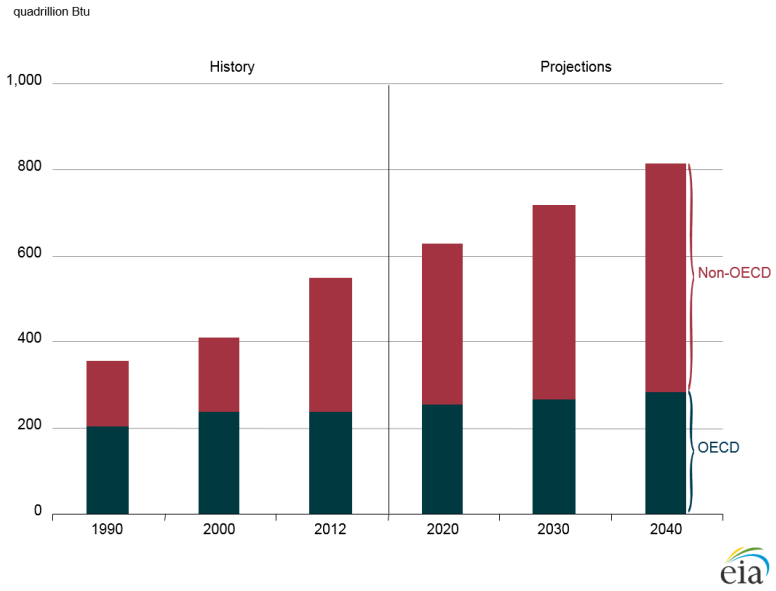

Figure 1.2: World energy consumption, 1990-2040. [5]

Furthermore, another advantage that benefits the expansion of renewable energy is the fact that its costs have dropped substantially in the past 30 years, whilst the price of fossil fuels have and are still fluctuating. In fact, fossil fuel and renewable energy prices, social and environmental costs are heading in opposite directions [6].

Yet another advantage of renewable energy sources is the fact that are less harmful to the environment, as they release way fewer greenhouse gases into the atmosphere. For that reason mainly, governments from all around the world have tried to reach an agreement for the application of a unified policy that could reduce CO_2 emissions by expanding the use of renewable energies.

All these advantages of renewable energy have produced a sustainable growth of its use in the last few decades. In Figure 1.3 is possible to observe an example of this, as from 2004 to 2014 the overall share of renewable energy in total final energy consumption has increased a 2.8 %.

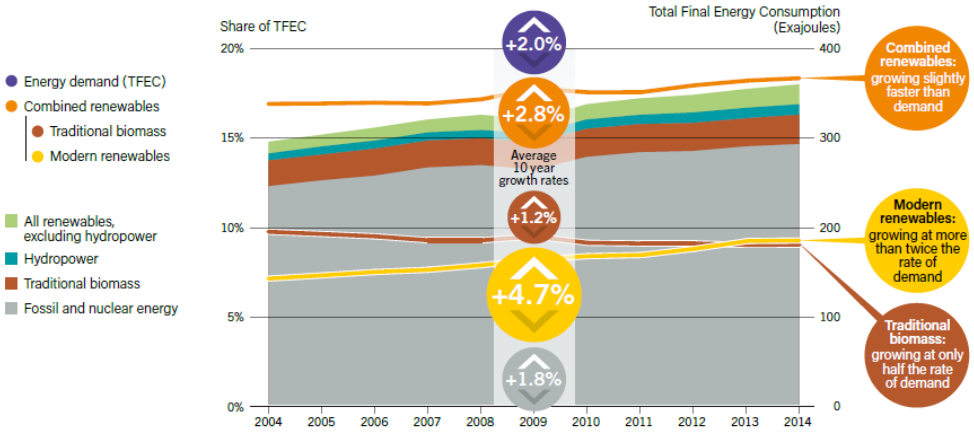

Figure 1.3: Growth in Global Renewable energy Compared to Total Final energy Consumption, 2004-2014. [7]

As shown in Figure 1.4, as of 2015 the share of final energy consumed corresponding to renewable energy reached a 19.3 %. If we break down this proportion, a 9.1 % was produced by traditional biomass energy while a 10.2 % was obtained from modern renewable technologies.

If we take only into account the production of electricity instead of the whole final energy consumption, it is easy to notice that renewable energies play a more important role. As shown in Figure 1.5, renewable energy sources correspond to a 24.3 % of the global electricity production. Breaking down again the share of renewables proves that hydropower is the most important type of renewable energy in the generation of power.

It is also important to highlight that Ocean energy corresponds only to a 0.4 % of the total. This is a surprising fact, taking into account that ocean energy potential resources are estimated approximately at 114040 TWh/year [8] and that the electricity produced globally in the year 2015 is estimated at 24107 TWh [9]. Given these facts, is clear that Ocean energy has enormous potential that needs to be researched.

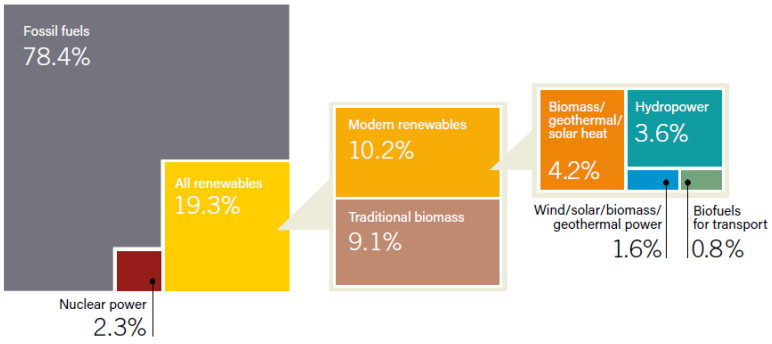

Figure 1.4: Renewable energy Share of Total Final energy Consumption, 2015. [7]

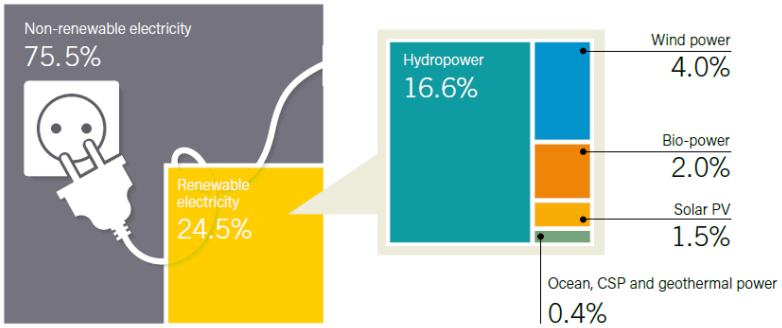

Figure 1.5: Renewable energy Share of Global Electricity Production, 2015. [7]

1.2 Ocean energy

Ocean energy refers to all the energy that can be harvested from the ocean. There are a few different possible technologies that can be used to extract this energy, the most important ones being [10]:

- **Tidal energy:** There are two ways to produce energy from tides. The first one consists in taking advantage of its potential energy, by building barrages in places like the mouths of estuaries to harness the energy of the tidal flow. Typically, the barrage only captures the energy of the water flowing out of the estuary from high to low tide. On the other hand, is possible to capture the kinetic energy of the tide by installing turbines that extract the energy of tidal currents. Tidal power has the distinct advantage of being highly predictable, compared to solar, wind, and wave energy. The regularity of the tides along with an immense energy potential helps make tidal energy development attractive [10].

- **Wave energy:**It is the energy that can be extracted from waves, which are mostly originated because of wind. Wave energy is considered one of the most promising renewable technologies. Compared with the other kind of renewable energies it is more dependable, as at a given site is available up to 90 % of the time, while solar and wind availability tend to just be of around 20–30 % of the time [10].

- **Ocean Thermal energy Conversion (OTEC):** OTEC uses the natural thermal gradient of the ocean to produce electricity. The heat stored in the warm surface water is used to heat a working fluid with a low boiling point with the objective of creating steam, which is taken to a turbine where is used to create electricity. After that, cold, deep water is pumped to the surface to re-condense the steam. This kind of ocean energy is only viable in the tropical seas, where the gradient is higher.

Regarding the development and expansion of this kind of technologies, at the end of 2016, there were only 536 MW of ocean energy operating capacity installed in the world. Of these, more than 90 % was represented by two tidal barrage facilities, the Sihwa plant in the Republic of Korea (completed in 2011) and the La Rance tidal power station in France (built in 1966). Aside from these two facilities, ocean energy technologies are still largely in pre-commercial development stages all around the world [7].

Of the 114040 TWh/year of ocean energy potential resources: 1200 TWh/year correspond to tidal energy, 29500 TWh/year to wave energy and 83340 TWh/year to OTEC [8].

It is important to notice that, although tidal energy is the one with less potential resources is also the most developed ocean energy technology. The previous implies that the development of new wave energy technologies has more potential at the end. Mainly because of two reasons, (a) there is a long way to go to have a reliable functioning technology, and (b) when is finally developed there is going to be more potential resources to be gathered.

1.3 Wave energy

As explained above, wave energy is the energy that can be extracted from waves. This type of energy has different advantages compared to other renewable sources [11].

- Sea waves energy density is the highest [12].
- Its environmental impact in use is limited [13].
- Its natural season variability is in line with the electricity demand in temperate climates [12].
- Waves are able to travel long distances without significant losses.
- As stated in Section 1.2, wave energy devices can produce energy 90 % of the time [10].

In Figure 1.6 the annual mean wave power density and predominant direction are shown. Wave energy is unevenly distributed around the world, mainly because of the different climates. As observed, the best wave zones are found in temperate zones, where European seas are located. Being this even another reason for the research on wave energy by European research groups.

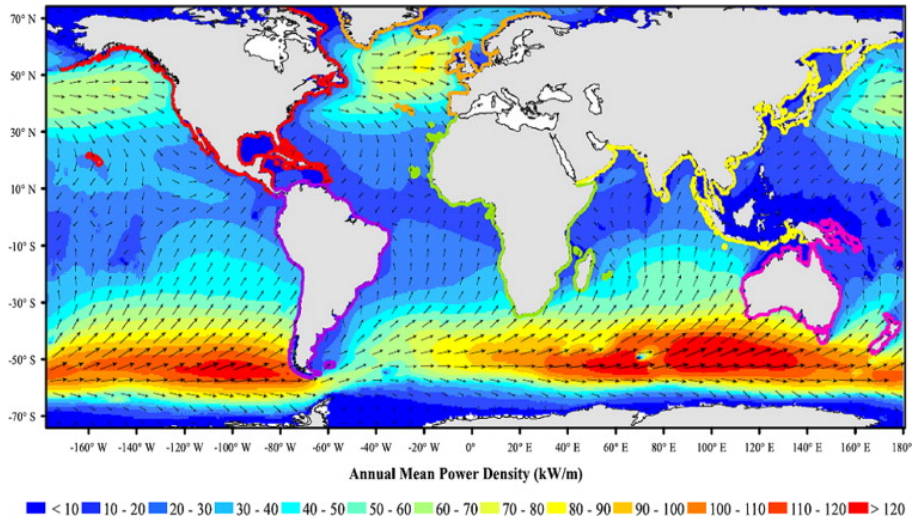

Figure 1.6: Annual mean wave power density and annual mean best direction. [14]

1.3.1 Wave energy Converters (WEC)

During this last period of history, when renewable energy research has widespread around the globe, many concepts of devices for the extraction of wave energy have been developed. Nevertheless, all these concepts can be classified depending on different criteria.

Location

If the criteria used to classify WECs is location, three different groups can be identified [11].

- **Shoreline devices:** When the device is located on or next to the shore. Thanks to the closeness to the coast, these devices are easy to maintain and are less likely to be damaged in extreme conditions. The downside is the fact that being located in shallow waters, incident waves have less power.
- **Nearshore devices:** In this case, the devices are usually attached to the seabed, which gives them a stationary base against which an oscillating body can operate. Being also situated in shallow waters they share the same disadvantage as shoreline devices, lower wave power.
- **Offshore devices:** Are situated several kilometers away from the coast, in deep water. This situation gives them the advantage of being able to harvest a greater amount of energy because of the highest energy content of deep water waves. On the other side of the coin, they have to survive more extreme conditions, maintenance is more difficult, and the connection to the grid is more complex.

Dimensions and relative position to the wave

If the criteria used for the classification of WECs is their dimensions and relative position to the wave, also three different groups can be separated [11].

- **Attenuator:** This kind of devices are situated in parallel to the wave direction and ride the waves. Its dimensions are similar to the wavelength.
- **Point absorber:** Its dimensions are relatively small compared to the incident wavelength, which makes wave direction not important. Point absorbers are distinguished by their heaving movement up and down. To obtain this behaviour, their structure can float on the surface of water or be submerged below the surface and rely on pressure differential.
- **Terminator:** This kind of device is characterised by having their principal axis perpendicular to the predominant wave direction in a way that it physically intercepts the waves. It has a similar dimension as attenuators.

Principle of operation

Within the classifications previously exposed, there is another way of classification possible, its principle of operation. One example of each of the most important broad type of wave energy converter in a more advanced development stage has been described briefly below [11, 15].

- **Submerged pressure differential:** Consists of a submerged point absorber that moves with the help of the pressure difference above the device between wave crests and troughs. It has two parts: one fixed to the sea bed containing a cylindrical chamber filled with air, and a movable upper cylinder. The variation of pressure of the air contained in the first part, produced by the variation of water on top of the device, is the origin of the heaving movement. One example is the Archimedes Wave Swing (see Figure 1.7).

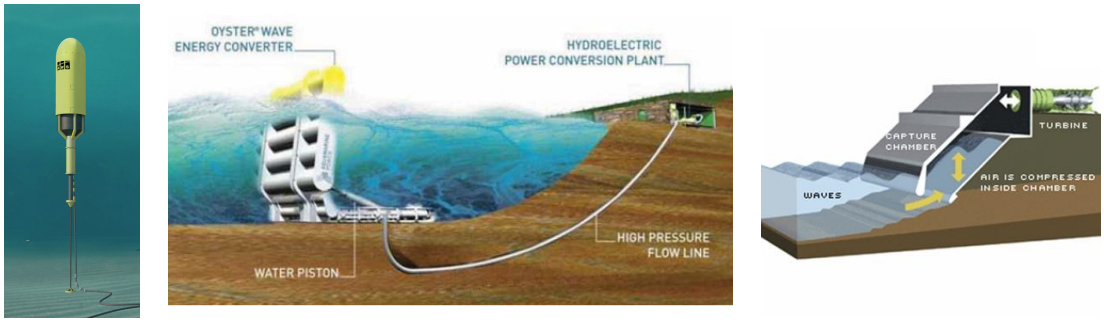

Figure 1.7: From left to right: Archimedes Wave Swing [16], Aquamarine Power Oyster [17], Wavegen Limpet [18].

- **Oscillating wave surge converter:** This kind of devices are normally composed by a hinged deflector that moves back and forth exploiting the horizontal velocity of the wave. They are terminator devices. The Aquamarine Power Oyster is an example (see Figure 1.7).
- **Oscillating water column:** It is constituted by a chamber with an opening to the sea below the waterline. When the wave approaches the construction, this wave enters the chamber and applies pressure to the air present inside. This air is then deployed into the atmosphere through a turbine that produces energy. When water retreats, the air circulates in the opposite direction through the same turbine, generating even more power. One example is the Wavegen Limpet (see Figure 1.7).
- **Overtopping device:** The idea behind this concept is that of a device that can gather water in a reservoir above the sea level, capturing it when waves overtop a barrier. This water is then released back into the sea passing through a turbine and so producing energy. One example is the Wave Dragon (see Figure 1.8).
- **Wave-activated bodies:** This last kind of wave power devices groups all those that capture the wave energy with floating bodies that oscillate with the wave. Pumps and generators are used to generate power. Two examples of this principle are two of the most advanced projects nowadays: (a) Ocean Power Technology's PowerBuoy, which is fixed onto the sea bed with a rope, and oscillates freely with the wave to move the fluid present in its Power Take Off (PTO) system; and (b) Pelamis, whose body reminds that of a snake, and produces energy with a PTO system on each joint powered with the oscillation of the wave (both can be seen in Figures 1.8 and 1.9, respectively).

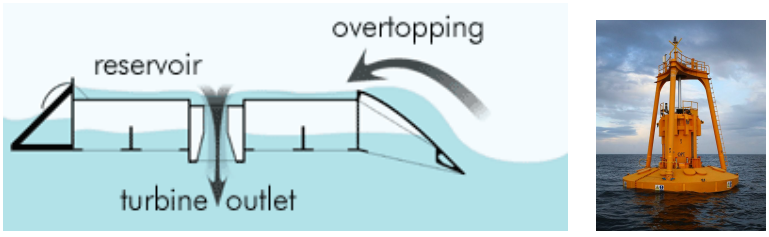

Figure 1.8: From left to right: Wave Dragon [19], OTP PowerBouy [20].

Figure 1.9: Pelamis [21].

1.4 Ocean Grazer (OG)

The Ocean Grazer project has taken one step beyond the research for a more adaptable and efficient WEC. Even more than that, since the Ocean Grazer platform is also able to generate electricity with other kinds of renewable energy technologies, such as wind and solar. Its structure will be located in deep water, more than 50 km offshore. And its core technology, contributing about 80 % of the energy generation, is a novel wave energy harvesting and storage device termed the multi-pump, multi-piston power take-off (MP^2PTO) system [22].

What makes the Ocean Grazer WEC stand out from other WECs are its flexibility and the possibility to store energy. Flexibility, thanks to the ability of the MP^2PTO system of adapting its operation to a wide range of waves, with an average extraction efficiency of about 90 % for waves ranging in height from 1 to 12 meters and periods of 4 to 20 seconds [23]. Regarding the storage of energy, the majority of existing WEC technologies transform the captured wave energy directly to electricity. On the contrary, the Ocean Grazer has the ability to store up to 800 MWh of loss-free energy before transforming it to power and supply it. This has the advantage of decoupling the electricity production from the availability of renewable wave energy [24, 25]

The system, as shown in Figure 1.10, is expected to reach a total height of 255 meters, 225 meters of which are submerged under the sea surface, and a diameter of up to 435 meters [22]. In relation to the energy output, the Ocean Grazer platform is expected to produce between 220 to 270 GWh/year, which would be enough to fulfil the electricity demand of around 70.000 households each year [22, 25].

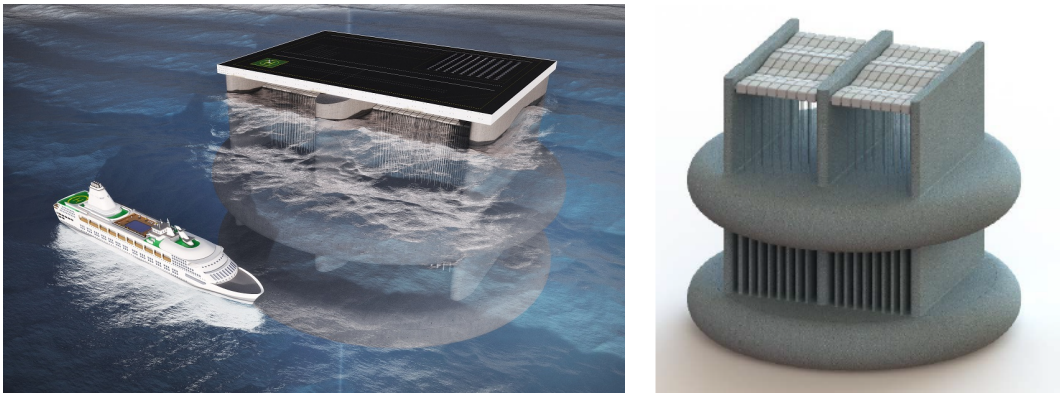

Figure 1.10: The Ocean Grazer platform submerged next to a cruise ship (left) and unsubmerged and without a top platform (right).

1.4.1 Multi-Piston Multi-Pump Power Take-off System

The main structure of the Ocean Grazer platform consists of two massive reservoirs containing a working fluid, such as water, circulating in a closed loop. As it is shown in Figure 1.11, the two reservoirs would be connected on the one hand by multiple multi-piston pumps that would transport the water from the lower to the upper reservoir (gaining hydraulic head, h); and, on the other hand, by various conducts that would release the water back to the lower reservoir via multiple turbines thus generating electricity.

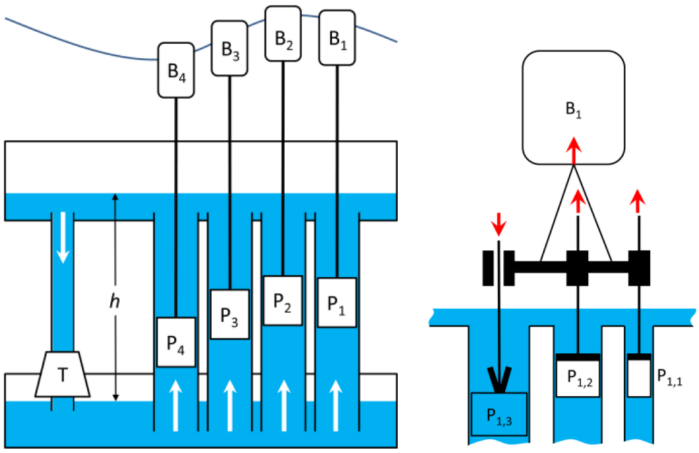

Figure 1.11: Schematic of the MP^2PTO system (left) and the multi-piston pump (right) [23].

The MP^2PTO system uses multiple floating buoys (B_i), interconnected in an array (termed as the *floater blanket*). Each of these buoys actuates one multi-piston pump (P_i) following the vertical displacement of an ocean wave. On the above figure only one floater blanket is represented; in reality, multiple parallel floater blankets are expected to be installed (see Figure 1.10).

The characteristics of the incoming waves vary with time. Moreover, the characteristics of the same wave will change as more energy is extracted along the floater blanket. Hence, the first 'equivalent' pump can potentially extract more energy than the following, and so on. Consequently, variable-load control for each of the pumps is needed to maximise the energy extraction in all situations. This will be possible through the configuration represented in the right-hand picture of Figure 1.11. The coupling between any buoy (B_i) and a number of variable-size pistons ($P_{i,j}$) can be controlled with a coupling mechanism that engages and disengages the pistons in order to adapt the pumping according to the incident wave [23, 26].

1.4.2 Current state

The Ocean Grazer project is still in its research phase. Up until this point, all research has been performed by the Ocean Grazer group within the Faculty of Science and Engineering of the University of Groningen.

In the current research stage, the project's main purpose is to deliver a solid proof of concept for the core technology of the Ocean Grazer WEC and platform, also known as technology readiness level 4 (TRL-4). Parallel to this, research into the economic, social, legal and environmental feasibility of the Ocean Grazer platform is performed as well [22].

With that objective, the team has been working on different numerical models that could explain the behaviour of the different parts of the system, with the purpose of helping in the development of the concept. Some examples of these modelled parts of the system are the pumping system, the turbine system, the floater blanket and the stability of the device in various wave conditions.

Furthermore, to try to validate those models, several functional prototypes of the major components of the MP^2PTO system have been developed. Two of the most important prototypes developed being (a) a prototype of a single multi-piston pump; and (b), a wave tank consisting of a 1/35th scale prototype version of the floater blanket with ten interconnected floater members, each equipped with an individual working and controllable pumping system to control and maximize the energy extraction.

1.5 Goals of this thesis

As explained before, recent work of the Ocean Grazer team has been oriented to a wide range of topics. One of the most recent works is the one developed by Jenifer Brenes [27]. In her thesis, she studied the behaviour of the wave tank experimental setup to try to determine the influence of the MP^2PTO system and floater blanket in the wave particle's motion and energy extraction.

To do so, she used the DPIV technique to calculate and study the kinetic energy present in various points along the floater blanket. This was carried under different scenarios to evaluate the efficiency of the prototype and the pumping configurations used.

During her research, a validation of the results was tried to be carried out, comparing the kinetic energy calculated with the DPIV measurements with the results of a CFD simulation. The comparison was not conclusive. Although trends of the results were in line, there was a discrepancy in their magnitude.

Consequently, a new take on the validation of results is needed, and here is where this thesis comes to play. Therefore, the main goals of this thesis can be summarised as:

1. Validate the experimental measurements previously taken of the wave energy extraction in the wave tank experimental setup. To succeed in that effort, a CFD simulation of the same experimental setup will be executed using a specialised program denominated ComFlow.
2. In case the new CFD results do not match, find the sources of discrepancy.

Chapter 2

Theoretical background

2.1 Water waves

To be able to estimate correctly the energy comprised in a portion of a wave first is necessary to understand exactly what a wave is and its behaviour. Surface elevation (η) and wave need to be distinguished. The previous refers to the instantaneous elevation of the sea surface relative to some reference level, while the latest corresponds to the profile of the surface elevation between two successive downward zero-crossings of the elevation or, alternatively, the profile between two successive upward zero-crossings [28].

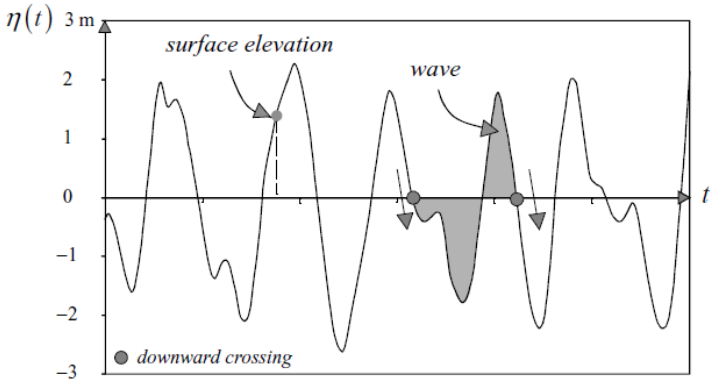

Figure 2.1: Definition of wave in a time record of the surface elevation [28].

Many models can be used to try to describe the evolution of the surface elevation in time and space. One of the most used, because of the simplicity that results of assuming linearity of the potential flow function, is the one related to the linear wave theory (see Section 2.2):

$$\eta(x,t) = a_0 cos(\omega t - kx) \tag{2.1}$$

where η (m) is the sea surface that depends on time t (s) and space x (m), a_0 (m) is the wave amplitude, ω (rad/s) the angular velocity depending on wave period T (s), and k (rad/m) is the wavenumber defined by the wavelength λ (m).

Consequently, as denoted, the main parameters that define a wave are its amplitude (a), its period (T) and its wavelength (L), represented in Figure 2.2. The amplitude corresponds to half of the wave height (H), the vertical distance between the highest and the lowest surface elevation in a wave. The time interval between the start and the end of the wave is what is known as the period of a wave. Finally, the wavelength is the distance between two successive peaks or two consecutive troughs.

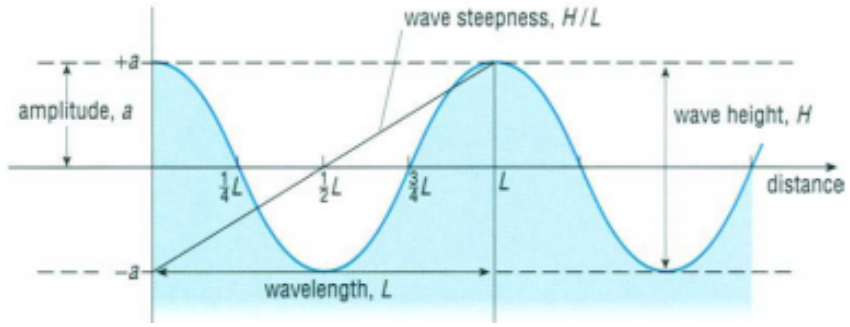

Figure 2.2: Vertical profile of two successive idealized ocean waves, showing their linear dimensions and sinusoidal shape [29].

Water waves can be originated by a wide range of phenomena, mainly wind but also earthquakes, floating structures or astronomical forces among others. All of them can be classified in terms of their period or wave length, as follows [28] (as represented in Figure 2.3):

- **Trans-tidal waves:** This kind of waves are the longest waves and have periods of more than one day. They are generated by low-frequency fluctuations in the Earth's crust and atmosphere.
- **Tides:** The periods of this category of waves can range from a few hours to somewhat more than a day and their wave lengths between a few hundred and a few thousand kilometres. They are generated by the interaction between the oceans on the one hand and the Moon and the Sun on the other.
- **Storm surges:** Their length and period are slightly shorter than that of tides. They are generated by the low atmospheric pressure and the high wind speeds in a storm.
- **Tidal waves:** Also known as 'tsunamis', are generated by a submarine earthquake. They are difficult to predict and barely noticeable in the open ocean.
- **Seiches:** Its frequency is normally equal to the resonance frequency of the basin in which they occur. They are generated by waves from the open sea, the source of which is not well understood.
- **Infra-gravity waves:** The period for this kind of waves consists typically of a few minutes. They are generated by groups of wind-generated waves and wind.
- **Wind-generated waves:** This category's wave periods are shorter than 30 s. As its name indicates, they are generated by wind. They can be classified in three subcategories:
  - Swells: They are one kind of surface gravity waves, the kind of waves that the Ocean Grazer aims to capture to extract its energy. They are denominated this way because they are dominated by gravity (periods longer than 1/4 s). When they leave the generation area, they take on a regular and long-crested appearance and are called swell.

– Wind sea waves: They are the other kind of surface gravity waves. In this case, though, they are generated by the local wind and are irregular and short-crested.

– Capillary waves: Their periods are shorter than 1/4 s. Instead of being affected by gravity, their main influence is surface tension.

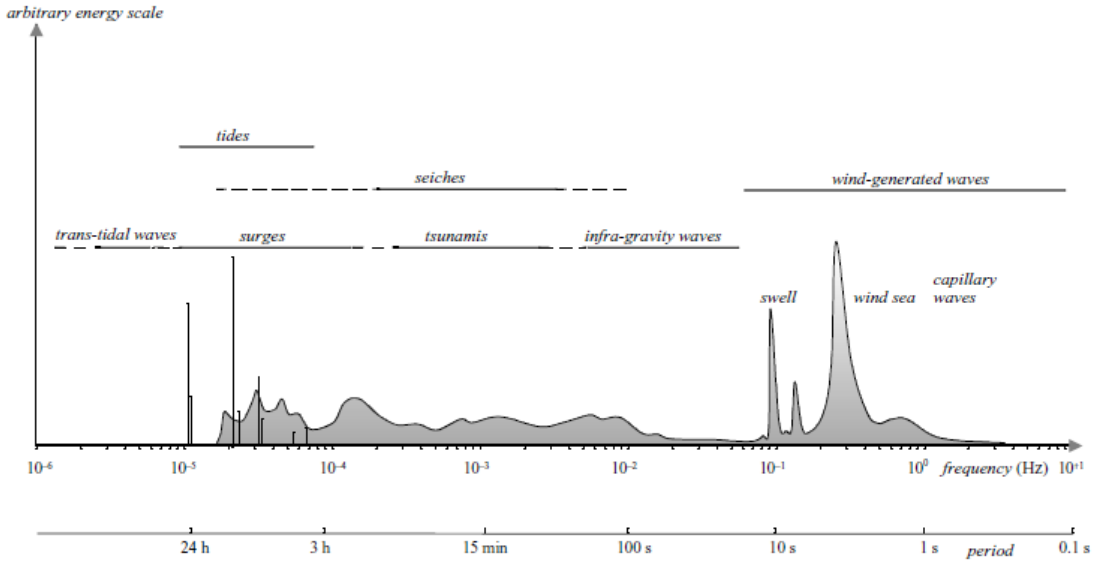

Figure 2.3: Frequencies and periods of the vertical motions of the ocean surface [28].

Furthermore, depending on the wavelength, waves can be classified in deep water and shallow water waves. This classification is necessary because simplified kinematic and dynamic formulation can be made [29]:

- **Deep water waves:** A wave is classified under this category when water depth (d, the distance from the surface to the seabed) exceeds half of the wavelength (λ), $d > \frac{1}{2}\lambda$.
- **Shallow water waves:** A wave is classified under this category when water depth (d, the distance from the surface to the seabed) is below a twentieth part of the wavelength (λ), $d < \frac{1}{20}\lambda$.

2.2 Wave particle motion

In deep water, water particles motion resembles almost a perfect circle. When a particle is situated at a wave crest, it moves in the same direction as wave propagation; whereas when the same particle is in a trough, it moves in the opposite direction. The diameter of said circle is equal to the wave height at the surface but, as the depth increases, the diameters decrease exponentially. However, below a depth roughly equal to half the wavelength the particles no longer move. Due to the exponential distribution of movement, almost 90 % of the energy is contained within a depth of a quarter of the wavelength [22]. In shallow water, where depth is less than half the wavelength and the waves 'feel' the seabed, the orbits become progressively flattened with depth. In Figure 2.4 a representation of the particle motion in deep and shallow water is shown.

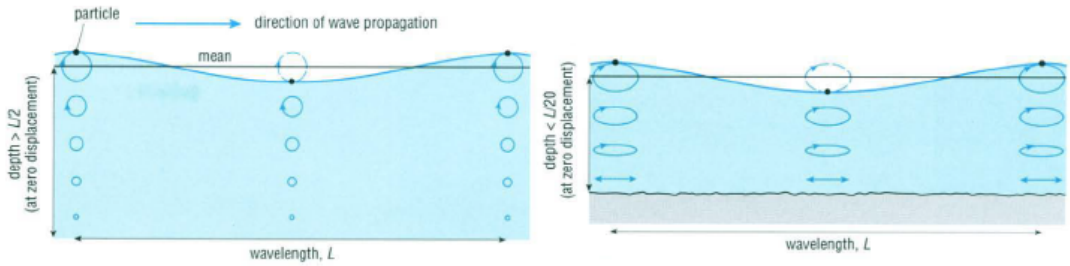

Figure 2.4: Particle motion in deep water waves (left) and shallow water waves (right) [29].

2.2.1 Linear wave theory

The linear theory of surface gravity waves has been the basic theory for ocean waves for about 150 years [28]. The only requirement to apply this theory is for wave's amplitude to be much smaller than its wavelength, which in the case of the waves aimed by the Ocean Grazer is true.

Idealisations

Several idealisations have to be made to develop this theory.

- Water is considered to be *ideal*. Which means that is assumed to be *incompressible*, to have *constant density*, and to have *no viscosity*.
- The water body must be *continuous*, with no air bubbles. In any other case, as when waves break, this modelisation is not valid.
- Water particles may never leave the surface nor penetrate the bottom.
- Water is only assumed to be subjected to the gravitational force.

Velocity potential function

The velocity potential function $\phi = \phi(x, y, z, t)$ is defined as a function of which the spatial derivatives are equal to the velocities of the water particles:

$$\phi(x, y, z, t) \text{ defined such that } u_x = \frac{\partial \phi}{\partial x} \text{ , } u_y = \frac{\partial \phi}{\partial y} \text{ and } u_z = \frac{\partial \phi}{\partial z} \tag{2.2}$$

Using this function that relates all velocities in all directions, is easier to obtain an analytical solution for the wave motion.

Balance equations and boundary conditions

Balance equations are the basis for the linear wave theory. Two balance equations are needed, a mass balance equation and a momentum balance equation. Equations (2.3) and (2.4) are the result of applying all the stated idealisations to said balances and using the velocity potential function. All the steps taken to obtain them can be reviewed by the reader in [28].

Laplace equation (From continuity equation): $$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{2.3}$$

Linearised Bernoulli equation (From momentum balance equations): $$\frac{\partial\phi}{\partial t} + \frac{p}{\rho} + gz = 0 \tag{2.4}$$

Furthermore, to obtain a valid analytical solution for the wave motion, determining the boundary conditions is key. In Figure 2.5 all the basic boundary conditions for the linear wave theory are represented. Lateral boundary conditions stand out because they reduce the wave to be described as *two-dimensional*, considering that waves only travel horizontally in the x-direction.

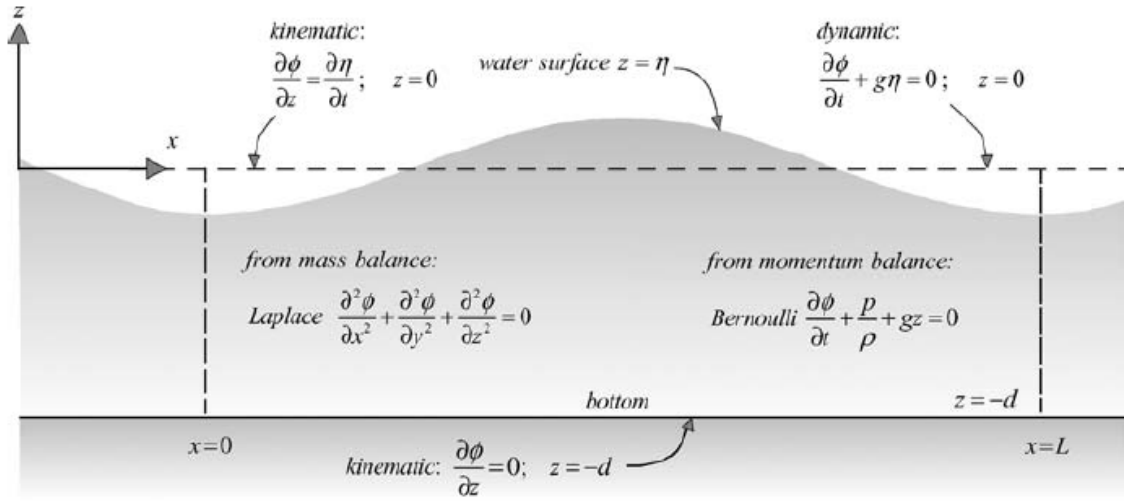

Figure 2.5: The basic equations and boundary conditions for the linear wave theory, in terms of the velocity potential [28].

Propagating harmonic wave. Kinematics

Considering all the previously introduced boundaries, an analytical solution of the Laplace equation can be obtained. This solution corresponds to the one that describes the evolution of the surface elevation in time and space, previously shown in (2.1). Resulting in the expression shown in (2.5) for the velocity potential function, namely,

$$\eta(x,t) = a_0\, cos(\omega t - kx) \tag{2.1 revisited}$$

$$\phi = \hat{\phi}\, cos(\omega t - kx) \quad \text{with} \quad \hat{\phi} = \frac{\omega a_0}{k}\frac{cosh[k(d+z)]}{sinh(kd)} \tag{2.5}$$

where $\omega\,(rad/s)$ is the the angular velocity, $a_0\,(m)$ is the amplitude of the wave, $k\,(rad/m)$ is the wavenumber, $d\,(m)$ is the depth, z (m) is the z-coordinate in relation to the water surface and d (m) is the water depth.

Consequently, taking into account the definition of velocity potential function shown previously, it is easy to obtain a mathematical expression for the velocities in the x and z-direction (as in the y-direction velocity is zero):

$$u_x = \hat{u}_x \, sin(\omega t - kx) \quad \text{with} \quad \hat{u}_x = \omega \, a_0 \frac{cosh[k(d+z)]}{sinh(kd)} \tag{2.6}$$

$$u_z = \hat{u}_z \, cos(\omega t - kx) \quad \text{with} \quad \hat{u}_z = \omega \, a_0 \frac{sinh[k(d+z)]}{sinh(kd)} \tag{2.7}$$

In the cases of deep water (when $kd \to \infty$) and shallow water (when $kd \to 0$), it is possible to simplify these expressions. In the case of deep water these expressions are represented in (2.8) and (2.9). For the shallow water's case, the new expressions correspond to (2.10) and (2.11).

Deep water ($kd \to \infty$):

$$u_x = \omega a e^{kz} \, sin(\omega t - kx) \tag{2.8}$$

$$u_z = \omega a e^{kz} \, cos(\omega t - kx) \tag{2.9}$$

Shallow water ($kd \to 0$):

$$u_x = \frac{\omega a}{kd} \, sin(\omega t - kx) \tag{2.10}$$

$$u_z = \omega a \left(1 + \frac{z}{d}\right) cos(\omega t - kx) \tag{2.11}$$

The previous deep water expressions, (2.8) and (2.9), show that the velocities decrease exponentially with the distance to the surface. On the contrary, the previous shallow water expressions, (2.10) and (2.11), show that the amplitude of the horizontal and the vertical velocities are constant and vary linearly along the vertical, respectively.

Regarding the path of the particles, it is easy to obtain them by integrating their velocities in time. A convenient approximation here is to consider a particle located near an arbitrarily chosen position (represented as $\bar{x}, \bar{z}$), and take the velocity at this location [28]. With local co-ordinates x' and z' (centered on $\bar{x}, \bar{z}$), particle relative paths are given by

$$x' = -a \, \frac{cosh[k(d+\bar{z})]}{sinh(kd)} \, cos(\omega t - k\bar{x}) \tag{2.12}$$

$$z' = a \, \frac{sinh[k(d+\bar{z})]}{sinh(kd)} \, sin(\omega t - kx) \tag{2.13}$$

Since the horizontal position x' varies as a cosine and the vertical position y' varies as a sine, each particle goes through an ellipse (equation (2.14)). The expressions of its horizontal and vertical semi-main axes are represented at (2.15) and (2.16).

$$\frac{x'^{\,2}}{A^{\,2}} + \frac{z'^{\,2}}{B^{\,2}} = 1 \tag{2.14}$$

$$A = a \, \frac{cosh[k(d+\bar{z})]}{sinh(kd)} \qquad \text{(horizontal semi-main axis)} \tag{2.15}$$

$$B = a \, \frac{sinh[k(d+\bar{z})]}{sinh(kd)} \qquad \text{(vertical semi-main axis)} \tag{2.16}$$

As in the case of velocities, several approximations of the previous expressions can be made for deep and shallow water. In the first case, the length of the two axes are equal, $A = B$. In the second case, the length of the axes can be calculated by

$$A = \frac{a}{kd} \qquad \text{(horizontal semi-main axis)} \tag{2.17}$$

$$B = a\left(1 + \frac{z}{d}\right) \qquad \text{(vertical semi-main axis)} \tag{2.18}$$

As the particles are closer to the bottom, their elliptical path grows flatter, $B \to 0$, and the ellipse degenerates into a straight horizontal line.

Dispersion relationship

The dispersion relationship shown below in (2.20) is an equation that relates different parameters of the wave, which has to be true for the wave to be a free wave. This relation is obtained by invoking the free-wave condition, making the surface pressure equal to 0. By substituting the harmonic surface profile (2.1) and the corresponding velocity atmospheric pressure (2.5) into the expression for this boundary condition (2.19) a relationship between radian frequency ω and wave number k is obtained.

$$\frac{\partial \phi}{\partial t} + g\eta = 0 \tag{2.19}$$

$$\omega^2 = gk\, tanh(kd) \tag{2.20}$$

As in the previous section, there are also simplifications of the expressions for deep-water and shallow-water waves. Equation (2.21) is the one related to deep water, where $tanh(kd) \to 1$ because $kd \to \infty$. Whilst (2.22) is the one related to shallow water, where $tanh(kd) \to kd$ because $kd \to 0$.

$$\omega = \sqrt{gk_0} \quad \text{where} \quad k_0 = k\, tanh(kd) \tag{2.21}$$

$$\omega = k\sqrt{gd} \tag{2.22}$$

2.3 Wave particle energy

Waves possess energy in two forms, potential and kinetic energy. In the first case, the fact that a wave is present at the water surface implies that water particles were moved from its original position to some other position. This means that some kind of work has been done against gravitation and this represents potential energy. In the second place, wave particles move, which represents kinetic energy [28].

In the case of the potential energy of a wave particle (E_p), is possible to calculate it with

$$E_p = mgh \tag{2.23}$$

where m (kg) is the mass of the water particle, g (m/s^2) the gravitational constant and h (m) the height of the particle.

Analogously, the kinetic energy of the same wave particle can be calculated as

$$E_k = \frac{1}{2}mv^2 \tag{2.24}$$

where m (kg) is the mass of the water particle, v (m/s) the velocity of the particle.

As stated before, the goal of the thesis is to compare the energy present in the experimental wave tank setup (as studied and estimated in Jenifer Brenes' thesis [27]) with the energy present in the CFD simulation of the same experimental setup. The study of the energy contained in a wave made in said thesis was centred on the kinetic energy present. Consequently, only the kinetic energy can be compared.

In this thesis, only the kinetic energy of waves will be calculated. The procedure of such calculation is explained in detail in chapter 5.

Chapter 3

ComFLOW software

ComFLOW is a program for the numerical simulation of fluid flow, based on the Navier-Stokes equations. The program has been developed by the University of Groningen. The ComFLOW program is one of the deliverables of the ComFLOW-3 project, which is the third installment of a series of projects aimed "To develop a user-friendly and validated numerical tool for marine and offshore industries to study complex free-surface problems, which is flexible in its application and has a coupling possibility to the other tools of participants" [30].

More precisely, the main objective of the ComFLOW-3 project is "To further improve, develop and validate the ComFLOW program for complex free-surface flows in the offshore industry and make it useable for advanced engineering applications by improved functionality and speed-up of the algorithms".

As part of the collaboration between the different departments of the Faculty of Science and Engineering of the University of Groningen, the ComFlow program has been selected to perform the CFD simulations of the Ocean Grazer's wave tank experimental setup for this thesis.

Figure 3.1: Logo of ComFlow [30].

3.1 Mathematics and numerics

The simulation that has been executed for this thesis is for a one-phase flow. In this section, a short review of mathematical and numerical one-phase model on which the COMFLOW program is based is given [30].

3.1.1 Mathematical Model for one-phase flow

Navier-Stokes equations

Fluid motion can be described by the Navier-Stokes equations. In this case, that water is considered as an incompressible and viscous fluid, the Navier-Stokes equations for a certain domain Ω can be simplified to the expressions (3.1) and (3.2), derived from the mass and momentum balance.

Conservation of mass:

$$\bigtriangledown \cdot \vec{u} = 0 \tag{3.1}$$

Conservation of momentum:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \bigtriangledown \vec{u} = -\frac{1}{\rho} \bigtriangledown p + \frac{\mu}{\rho} \bigtriangledown \cdot \bigtriangledown \vec{u} + F \tag{3.2}$$

where $\vec{u}$ (m/s) is the velocity vector, p (Pa) the pressure, μ $(Pa \cdot s)$ the dynamic viscosity, ρ (kg/m^3) the density and F (N) the external forces.

Boundary conditions and free surface

Like in the case of linear wave theory, boundary conditions at the boundary $\partial\Omega$ and the surface are needed to solve the Navier-Stokes equations. At solid boundaries, the fluid is prevented from going through and is supposed to stick to the wall because of viscosity. This is achieved by imposing $u = 0$ at solid boundaries of the domain and solid objects.

Regarding the free surface, its displacement is described with (3.3). Equations (3.4) and (3.5) correspond to the boundary conditions of the free surface, and are obtained from the normal and tangential stresses continuity.

$$\frac{Ds}{Dt} = \frac{\partial s}{\partial t} + (\vec{u} \cdot \bigtriangledown) s = 0 \tag{3.3}$$

$$-p + 2\mu \frac{\partial u_n}{\partial n} = -p_0 + 2\sigma H \tag{3.4}$$

$$\mu \left(\frac{\partial u_n}{\partial t} + \frac{\partial u_t}{\partial n} \right) = 0 \tag{3.5}$$

where $s(x,t) = 0$ gives the position of the free surface, u_n (m/s) is the normal component of velocity, u_t (m/s) is the tangential component of velocity, p_0 (Pa) is the atmospheric pressure, σ (N/m) is the surface tension and $2H$ (m^{-1}) denotes the total curvature.

Calculation of forces

The fluid induces a force on an object in the domain. This force normally would have two components, the pressure force and the shear force. This last component is neglected because is much smaller than the pressure force, resulting in equation 3.6.

$$F_p = \int_S p\, n\, dS \tag{3.6}$$

where S is the surface of the object.

3.1.2 Numerical model

Cell labelling

To perform the simulations, the flow domain Ω is divided into cells forming a grid with staggered variables. There are cells with different characteristics. This fact is incorporated into the numerical method by introducing edge and volume apertures. These apertures allow determining if the cell face or volume is open to flow. Based on this, cells are given geometry labels that describe which kind of cell it is: a fluid, a boundary (B) or an exterior cell (X). Furthermore, to describe the free surface other labels are used depending on if the cell is empty (E), is a surface cell (S) or if it is a fluid cell (F). The correct mathematical formulation for each cell is chosen depending on these labels.

Discretisation and solution method

Navier-Stokes equations are discretised in time and space. For time discretisation, the Euler (first-order accurate) or the Adams-Bashforth method (second-order accurate) can be used. Regarding spatial discretisation, (second-order) central discretisation and first or second order upwind discretisation can be chosen.

To solve the discretised one-phase Navier-Stokes equations, first at each time step a Poisson equation for the pressure is solved using SOR-iteration with an automatically adjusted relaxation parameter. When a solution is found, the new velocity field is computed, and subsequently, the free surface is displaced using the VOF-method combined with a local height function [30].

Finally, the time step for the next iteration is adjusted using the CFL-condition, which depends on the CFL-number calculated with (3.7). If this number is greater than an arbitrarily stipulated number the time step is halved, and if it is smaller than another specified number during ten successive time steps, the time step is doubled.

$$CFL = \max_{i,j,k} \left(\frac{|u_{ijk}|\,\partial t}{h_{x,i}} + \frac{|v_{ijk}|\,\partial t}{h_{y,j}} + \frac{|w_{ijk}|\,\partial t}{h_{z,k}} \right) \tag{3.7}$$

where u, v and w are velocity components; h_x, h_y and h_z denote mesh sizes in the corresponding directions; and i, j and k correspond to the cell number in the x, y and z directions, respectively.

3.2 Simulation execution procedure

The simulation execution procedure varies depending on the type of simulation to be run. Some examples of the variables that can influence on the steps to follow and the input needed are: if there are moving bodies or not, if the simulation is of a one-phase or two-phase flow, if there is an incoming wave predefined or not, on the complexity of the liquid distribution, among others. Nevertheless, the simulation procedure is more or less the same for all simulation types, varying only the number of steps to take.

Input files

There are two mandatory input files: *geometry.in* and *comflow.in*. The first one contains the information related to the geometry of the simulation. While the rest of basic information (like configuring the domain, time parameters, selection of mathematical methods, grid configuration, physical parameters, post-processing options, etc.) has to be included in the *comflow.in* file.

Alternatively, some of the information related to this last file (especially grid domain and post-processing options) can be entered in a XML-style file named *comflow.cfi*. It is also the case of the files *grid.in* and *liquid.in*, where grid information and a more complex liquid distribution can be input, respectively.

Furthermore, if a certain object is supposed to move, the motion of the body can be configured in a file denominated *motionobject.in*. Additionally, this movement can be interactive with the rest of bodies and liquid. In this case, additional files are needed to configure the interaction (mass.in, *spring.in*, *damping.in* or *external_force.in* are examples).

Procedure

Once all the input files are configured, the execution process of the simulation has the following steps.

1. In the first place, the GEODEF program has to be run. With its execution, the definition of the geometry is obtained in an additional file denominated *apertures.in*. In the case that a moving body is present, a file denominated *geomoving.in* is also output. These files are needed for the execution of the simulation program itself, ComFLOW.
2. (Optional) Then, if the liquid distribution has been configured in an external file (*liquid.in*), is the turn of LIQDEF. This program is run for the definition of the initial fluid configuration. The result of its execution is a file named *liquid_distr.in* also needed for ComFLOW.
3. Finally, the simulation program ComFLOW is executed. It can deliver different output depending on the post-processing parameters configured (see Section 3.3).

In Figure 3.2, a schematic of this procedure can be seen.

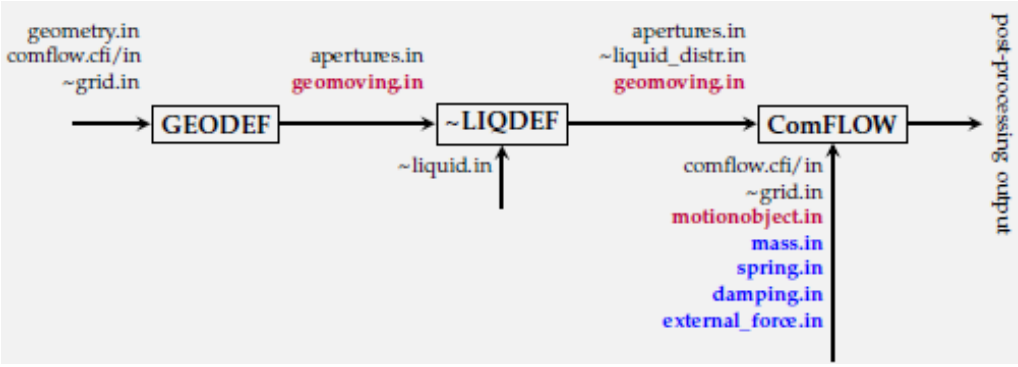

Figure 3.2: Calling sequence of the pre-processors and ComFLOW (fixed objects). Optional steps or input files are indicated by a "∼" symbol. In red, files necessary for the configuration of moving bodies. In blue, files necessary to configure interactive body motion [30].

3.3 Post-processing

ComFLOW gives the possibility to extract simulation results in a wide range of different forms. In this Section, the only two of ComFLOW's post-processing features used to extract result data for this thesis, *Snapshots* and *special boxes*, will be explained. For the rest of features (like *fill boxes, stream lines, particle paths,* among others) the reader is referred to ComFLOW's user manual [30].

3.3.1 Snapshot

One of the primary post-processing options in ComFLOW is the visualisation of 2-D and 3-D snapshots. This visualisation is possible thanks to a Matlab extension specially designed for this.

In Figure 3.3, ComFLOW's Matlab user interface can be seen. Different kind of information can be visualised, such as absolute velocity, velocity components, pressure or vorticity, among others. This information can be plotted in 2-D or 3-D mode. In the 2-D mode, only one slice of the domain is shown (in the XZ, ZY or YZ planes) while on the 3-D mode information is represented in the X, Y and Z axis. In this last case, there is also available the option of plotting a slice (with a 2D plot) in the desired coordinates. This slices can contain various types of 2-D objects, a color plot or a contour plot. An example of a plotted snapshot can be seen in Figure 3.4.

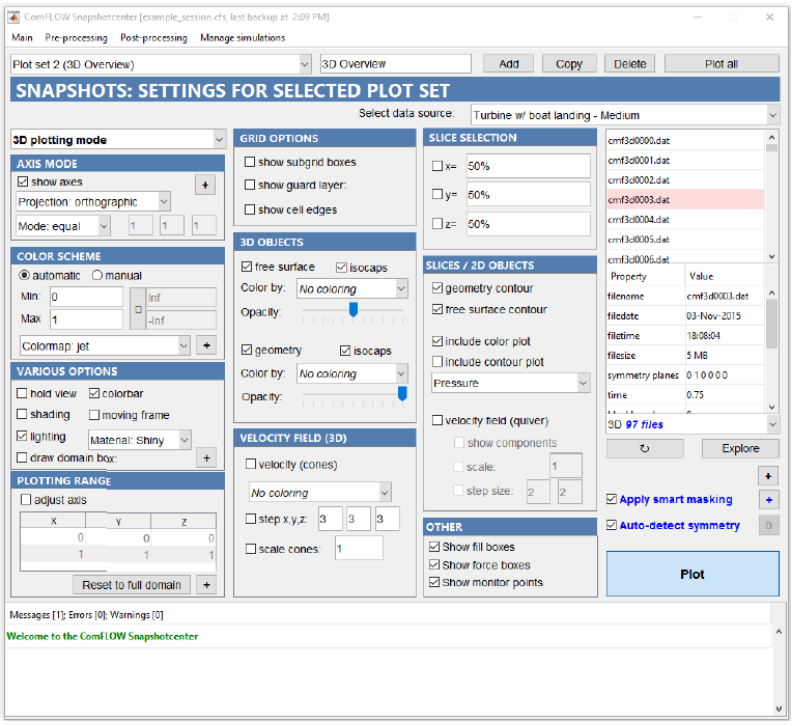

Figure 3.3: The snapshot settings window [30].

All the information necessary to represent all these plots is stored in different files with a *.dat* extension. Each of this files contains the information for one specific time step. The number of files, as well as the interval of time between them, can be configured in the *comflow.in* or *comflow.cfi* files. Among the information stored in these snapshot files, there is the water height for the whole domain. This is the information used from the snapshots files for this thesis. The code used to extract this information is explained in Chapter 5.

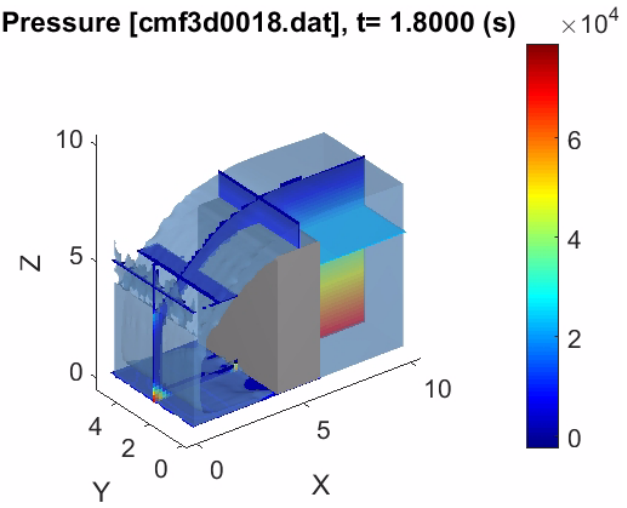

Figure 3.4: A snapshot.

3.3.2 Special box

Special boxes can be seen as a very basic snapshot files that contain limited amounts of information. This feature is used to store data of the pressure, velocities and geometry/liquid filling ratios for a small specified part of the domain. In this case, in contrast with snapshots files, a file for each time step is stored.

Boxes can be defined such that for all cell centres enclosed by the box, the pressure (p), liquid fraction and velocity components (u,v,w) are written to a file. The number of boxes (with a maximum of 999 boxes) is defined by the parameter nrboxes in the *comflow.in* file. To prevent unnecessary data storage, a start and end point in time can be provided for each box [30].

In this thesis, a special box for each analysed zone of the wave tank has been defined, with a total of seven boxes along the tank. This feature has been used to extract the velocities of these zones to calculate the kinetic energy present in them. The code used to extract this information is explained in Chapter 5.

Chapter 4

Water tank experimental setup

As expressed in Section 1.4.2, with the objective of validating the core MP^2PTO technology, a wave tank consisting of a 1/35th scale prototype version of the floater blanket with ten interconnected floater members has been constructed. This is the experimental setup studied on Jenifer Brenes' thesis [27] as well as on this thesis.

The 1:35 scale model of the designed MP2PTO system is shown in Figure 4.1. The flap paddle that generates the waves as well as the engine that drives it, both can be seen in the right-side picture. This generating unit is situated on the right side of the wave tank as seen in the left-side picture. The engine has two parameters that can be manually set, the rotating arm with a maximum value of 0.25 m, and the frequency which can be set to a maximum of 60 Hz [31].

The beach is situated on the left part of the wave tank as seen in the left-side picture of Figure 4.1. Its mission is to absorb the waves in order to reduce the amount of reflection. Moreover, with the goal of minimizing the effect that the remaining reflection has on the energy extraction, a plate is installed at the back of the floater blanket.

The setup also contains ten pistons with each one attached to a different floater blanket member. These pistons can be seen on the left-side picture of Figure 4.1. Their main mission is to pump water between two reservoirs in order to store potential energy by creating a hydraulic head.

Figure 4.1: Wave tank experimental setup.

A technical drawing of the wave tank experimental setup geometry is shown in Figure 4.2. The wave tank is 1.20 m high, has a width of 0.77 m and a length of almost 10 m. The wave flap is separated from the right side of the tank by 0.5 m. Moreover, the beach begins 2.72 m away from the left side of the tank. Between these two points, the floater blanket is situated. Finally, the water level in the tank is usually at 0.90 m. All this data has been used to define the geometry of the ComFLOW simulation as well as the liquid distribution.

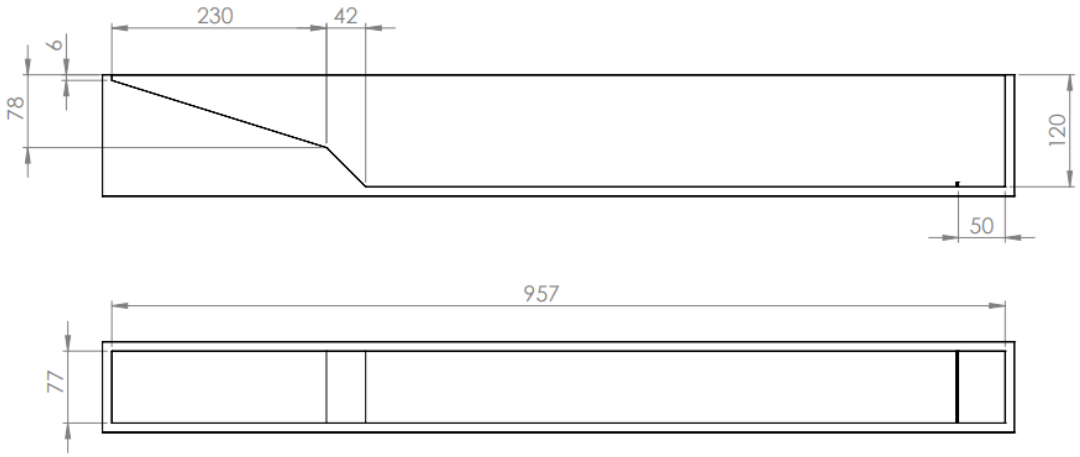

Figure 4.2: Wave tank experimental setup geometry (Dimensions in mm).

4.1 Experimental measurements

As commented, Jenifer Brenes made the experimental measurements to be validated. In this section, a summary of the procedure she followed to take the measurements, as stated in her thesis [27], is presented.

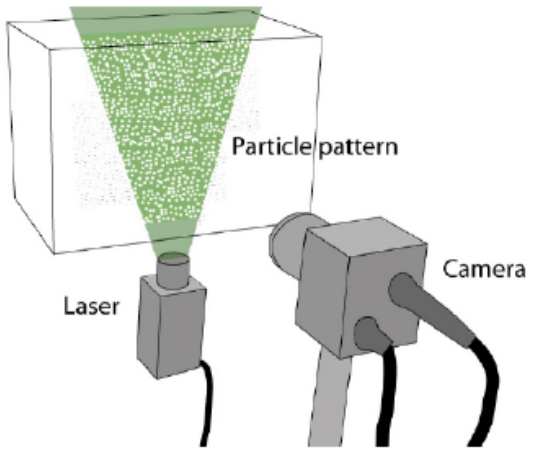

Figure 4.3: Principle of DPIV [32].

The experimental measurements were carried out using the Digital Particle Image Velocimetry (DPIV) technique [32, 33]. In these experiments, a high-definition camera was used to record the movement of free particles introduced in the wave tank. The experiments were carried out in a dark room, with the wave tank illuminated only by a laser sheet and the camera positioned parallel to this illuminated sheet, such that it could capture the particles' motion (see Figure 4.3).

Once a video was obtained, a frame by frame analysis was carried out that resulted in a velocity field for each pair of frames. Finally, this fields could be used to calculate the kinetic energy present in the recorded area for the duration of the video.

The camera and the laser were subjected by a special rail system that made the movement of the camera/laser pair easier, and the distance between the pair and the floater blanket always constant. To achieve this, the rail system was installed in parallel to the floater blanket and the windows shown in Figure 4.1.

The camera only could capture small areas below the floater blanket. Consequently, to be able to study the kinetic energy below all the floater blanket, measurements were taken area by area. Seven areas distributed along the wave tank were studied in total. In Figure 4.4 a sketch of the floater blanket is shown with this seven areas represented in yellow boxes. The areas are numbered from 0 to 6 from the right (the part of the floater closer to the wave flap) to the left (closer to the end of the tank).

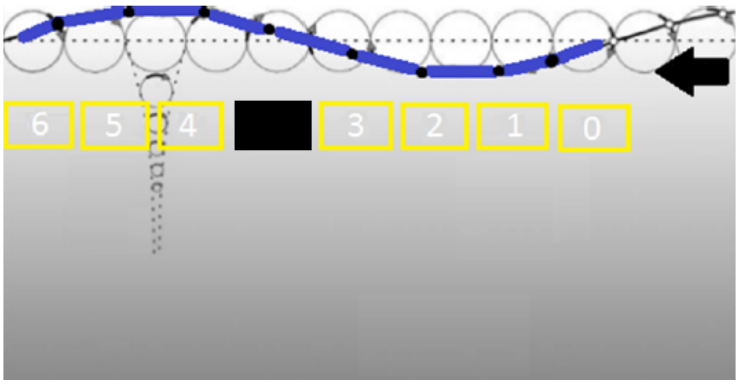

Figure 4.4: Division of areas measured under the floater blanket [27].

The experiments for the measurements were carried out in 5 different scenarios:

1. **No FB + No PTO:** In this case neither the floater blanket nor the pumping system were present in the tank.
2. **FB:** In this second scenario only the floater blanket was present. This allows studying the energy needed to move it.
3. **Set1M:** In this configuration the floater blanket and the pumping system were present. Additionally, the settings are configured according to one of the proposals which is considered as an optimal one.
4. **Set2W:** This is similar to the last one with the difference that another optimal configuration is used.
5. **Set1M + No final plate:** Similar to the third scenario. The only difference being that the final plate in the wave tank is taken out.

In addition, the measurements were taken with two different wave profiles with the aim of obtaining different results depending on the incident wave. Table 4.1 contains all the parameters for each wave profile.

Table 4.1: Wave profiles used in the experiments [27].

Wave Profile	High Wave	Small Wave
Wave Height [cm]	8	6
Wave Frecuency [Hz]	0.62	0.74
Arm Length of the motor [cm]	10	5
Machine Frecuency [Hz]	34	40
Wave Period [s]	1.62	1.36

4.2 ComFLOW simulation parameters

In this thesis, 2-D simulations of only one of the experimental scenarios were carried out to validate the measured results. The simulated case was the one where neither the floater blanket nor the pumping system were present in the tank (No FB + No PTO). That is because in the version of ComFLOW used to execute the simulations, the floater blanket was not possible to simulate. Furthermore, only one wave profile was taken into account, the 'High Wave' profile in Table 4.1. Finally, the computer used to carry out the simulations is an Intel core i5 @ 3.2 GHz with a 32 GB of RAM.

All the files used to configure the simulation with the information presented below can be checked in Appendix A.

Geometry

The schematic of the wave tank represented in Figure 4.2 was used to configure the geometry of the tank for the ComFLOW simulations in the file *geometry.in*.

Wave paddle motion

The motion of the wave paddle was configured according to (4.1), which describes the angle of the paddle in relation to the vertical direction in the case of the 'High Wave' profile.

$$\alpha = 4 \sin\left(t \frac{2\pi}{1.62}\right) + 3.4 \tag{4.1}$$

being α (°) the angle of the paddle and t (s) the time.

This equation includes much information regarding the wave paddle motion: (a) it has an amplitude of oscillation of 4°; (b) the same period as the wave profile (1.62 seconds); and, (c) the oscillation is not centered in the vertical direction, as shown in Figure 4.5.

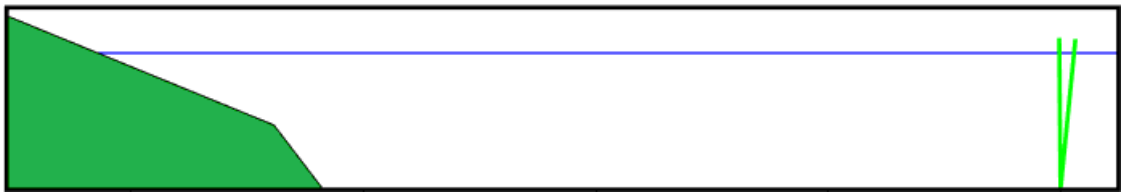

Figure 4.5: Wave tank schematic with the two limits of the wave paddle oscillation represented in green.

Other simulation parameters

Furthermore, other parameters have been considered among which the following stand out.

- The considered domain has a width of 9.57 m and a height of 1.2 m, just like the size of the whole wave tank.
- In relation to the boundary conditions applied, a Neumann condition is configured to all the walls of the domain by imposing a boundary flow velocity of zero.
- Regarding the initial liquid configuration and its physical properties, the same characteristics taken into account in the experiments have been considered. A depth of 895 mm and a density of 1027 kg/m^3 [27].
- In the case of the numerical parameters, Euler's method was used to integrate in time while a second order upwind discretisation was selected for the spatial discretisation solving.
- Finally, with reference to time parameters, time step has been adjusted with the CFL-condition procedure explained in Section 3.1.2, and all simulations have a duration of 50 seconds.

As mentioned before, all these parameters and many more can be seen in the configuration files presented in Appendix A.

Areas of study

As in the case of the experimental measurements, the kinetic energy content was analysed in seven different areas distributed along the tank, the same areas as in the experimental measurements. The exact coordinates and size had to be determined based on the limited available information from the experimental measurements.

Regarding its size, the areas were determined to have a width of 10.90 cm and a height of 18.85 cm, based on the result files obtained from the DPIV analysis in [27]. The coordinates of the bottom-right corner for each one of these areas are shown in Table 4.2, where the x-axis originates in the wave flap in direction to the beach and the z-axis originates on the water surface in direction to the sky. A graphical representation of these areas can be seen in Figure 4.6.

To obtain the velocities in these areas, a *special box* of a slightly bigger size was configured for each one of the areas in the *comflow.in* file. Afterwards, during the post-processing, the velocities of only the exact area of 10.90x18.85 cm are considered to calculate its kinetic energy. These calculations are explained in detail in Chapter 5.

Table 4.2: Coordinates of the bottom-right corner for each of the studied areas.

	area 0	area 1	area 2	area 3	area 4	area 5	area 6
X [m]	2.811	3.111	3.411	3.711	4.311	4.611	4.911
Z [m]	-0.278	-0.278	-0.278	-0.278	-0.278	-0.278	-0.278

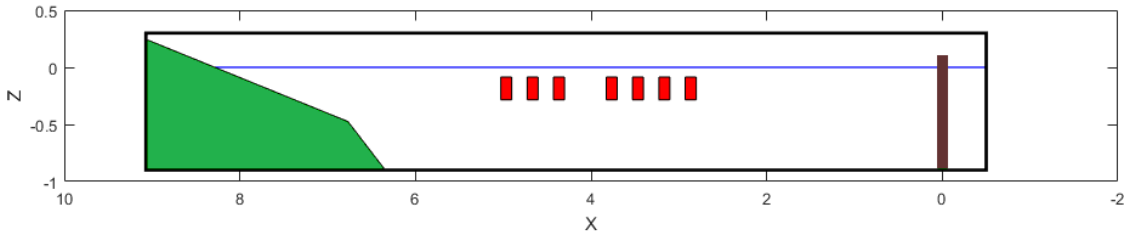

Figure 4.6: Wave tank schematic with the studied areas represented in red.

Cases of study

To solve a CFD simulation the considered domain has to be divided into cells. This is what is known as the mesh or grid. For this thesis, five different meshes have been used to solve the problem with a refinement ratio of 1.5. In other words, the problem has been solved five times with a different mesh in each case. Consequently, five cases of study have been considered. In Table 4.3 the number of cells in each direction for each one of the five cases of study is expressed.

Table 4.3: Number of cells in each direction for each case of study. Refinement ratio of 1.5.

Case of study	Number of Cells	
	X-direction	Z-direction
Mesh 1	304	40
Mesh 2	456	60
Mesh 3	684	90
Mesh 4	1026	135
Mesh 5	1539	202

Solving the same problem with different refined meshes gives the opportunity to analyse the influence of space discretisation on the results. The process of resolving a model with finer and finer meshes and comparing the results between these meshes is known as mesh refinement. This way, it is possible to judge the convergence of the solution with respect to mesh refinement.

The changes in the solution between meshes become smaller as the mesh is more refined following an asymptotic behaviour. Eventually, these changes will be small enough that the model can be considered to have converged [34]. This is the original motive behind the execution of the simulation for this five cases of study, to determine the convergence of the solution.

Finally, is important to mention that the configuration files in Appendix A correspond to only one of the cases of study, more specifically of Mesh 1. For all the other cases the only variables that change are the ones that define the number of cells in each direction. This variables are *imax* and *kmax* under the *grid parameters* section of the *comflow.in* file.

Chapter 5

Post processing of results

For each simulation, ComFLOW stores a great quantity of output data in the form of multiple files. This data has to be correctly processed to obtain the desired information in the most easily accessible form possible.

Originally, the aimed information to be obtained in this thesis was the kinetic energy contained in each one of the areas of study previously presented. To do so, the velocities of the particles inside those areas had to be gathered. As previously explained, this was possible by configuring a Special Box for each area in the *comflow.in* configuration file. This way, all velocity data would be stored in multiple special box files (almost one file per simulated time step and special box).

As it will be explained in Chapter 6, after carrying out the analysis of the obtained kinetic energy results, a frequency domain analysis of the water height of the water's surface was done. Thus being able to detect and study possible physical phenomena and the amount of reflection present in the tank. Consequently, the water height of the tank's surface had to be gathered. As mentioned, this was possible by configuring the output of Snapshot files in the *comflow.in* configuration file. Therefore, all water height data would be stored in multiple snapshot files (the number of output files is configured in the same configuration file, and are equally spread over the simulated time).

For each simulation case, these special box and snapshot files are stored in the *datasb* and *data* folders of the simulation's directory, respectively. Figure 5.1 gives an idea of the most important folders present inside each studied case (each mesh simulated).

In the following sections, all the coded procedures for the calculation of kinetic energy (Section 5.1) as well as the frequency analysis of the water height (Section 5.2) are explained.

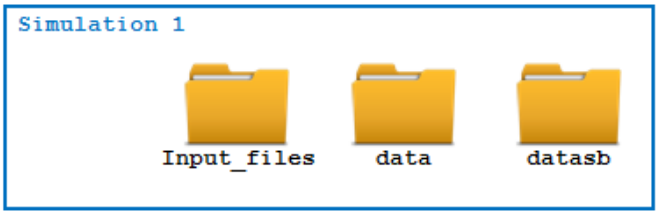

Figure 5.1: Most important folders present in each ComFLOW's simulation directory.

5.1 Kinetic energy

The procedure to calculate the kinetic energy contained in each of the studied areas has been divided into two Matlab functions.

On the one hand, the function coded in the Matlab file *read_velocity_parallel.m* (see code in Appendix B.1) is used to read all Special Box output files stored in one *datasb* folder and return the information stored in those in the form of two cell arrays variables named *coord_tot* and *vel_tot*.

On the other hand, the function coded in the Matlab file *cal_Ek_parallel.m* (see code in Appendix B.2) is used to calculate the kinetic energy at every time step in all the areas of study. Its input are the *coord_tot* and *vel_tot* variables output by *read_velocity_parallel*, which correspond to the mesh cell coordinates and its associated velocity, respectively. After the calculations, three new cell array variables named *Ek_tot_t*, *Ek_tot_x* and *Ek_tot_z*, which correspond to the absolute kinetic energy and its x-axis and z-axis components for all the studied areas, respectively.

Furthermore, an additional function was programmed in Matlab file *compare_windows.m* (see code in Appendix B.3). This function has as an input the absolute, x-axis or z-axis kinetic energy of different studied cases (the results of the simulation for different meshes). Then, it compares them between them as well as with the maximum and minimum DPIV measured values obtained from [27] for each of the studied areas by representing all this information in the same plot. The names to be plotted in the plot legend also has to be input with variable *E_names*.

In the following pages, all these functions will be further explained. In Figure 5.2 a block diagram of the kinetic energy calculation and comparison process is shown.

5.1.1 *read_velocity_parallel* function

As previously stated, this function is used to read all Special Box output files saved in one *datasb* folder and return the information stored in those in the form of two cell arrays variables named *coord_tot* and *vel_tot*.

In the simulations carried out in the present study, seven Special Boxes have been configured, one for each studied area. Each special box is configured in the *comflow.in* configuration file to be slightly bigger than the area to study.

Furthermore, if the configured coordinates of the Special Box do not match exactly with the limits of the cells of the mesh used to carry out the simulation calculations, the size of that Special Box increases even more until it matches. A schematic representing this situation is shown in Figure 5.3.

As commented before, in the *datasb* folder of the simulated case there are as much Special Box files as simulated time steps for each configured Special Box (if a simulation has five time steps and two boxes, in the corresponding *datasb* folder there would be ten files). In each one of these files, the coordinates of all the limits of the mesh cells contained inside the area of the special box (the cells in red in Figure 5.3), as well as the velocity components at the center of these cells are stored.

Consequently, this is the information that the currently explained function extracts from the files and stores in Matlab's cell array variables.

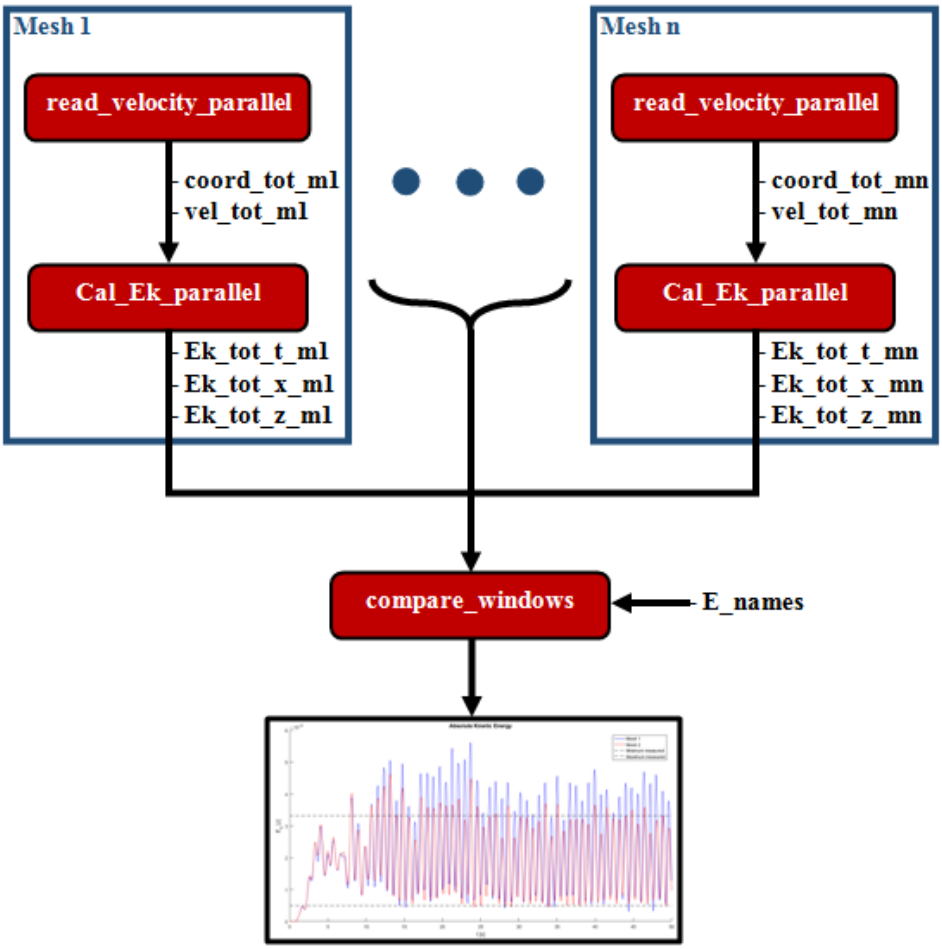

Figure 5.2: Kinetic energy calculation procedure for multiple simulation cases (mesh 1 to n).

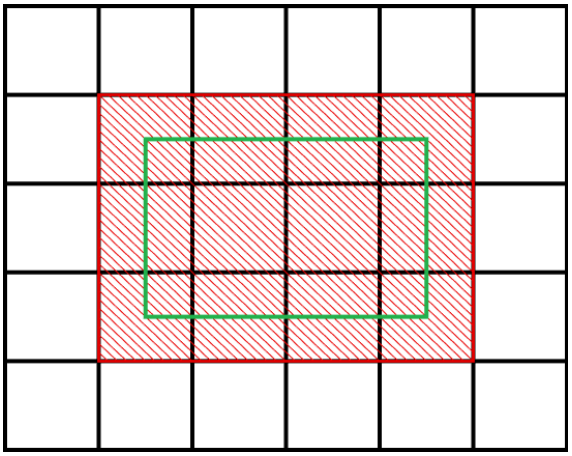

Figure 5.3: Example of simulation mesh (black), configured Special Box area (green) and real Special Box area (red).

Input

- **num_boxes**: Corresponds to the number of Special Boxes for which information is stored in the selected 'datasb' folder (in this case 7). It is used to calculate the number of files per Special Box present in said folder.

Output

- **coord_tot** [m]: Is a cell array variable. Its length is equal to the number of Boxes which have information stored in the selected *datasb* folder. Each Special Box has a cell of the array assigned (Cell number one corresponds to Special Box number one, etc.).

  In each cell, another cell array of length three is stored. In each one of this new cells, an array is stored that corresponds to the X, Y and Z coordinates of the mesh cells inside that Special Box, in that order.

  For example:

  - coord_tot{2}{3} would correspond to an array of the z-coordinates of the mesh inside Special Box number 2, ordered from smallest to highest coordinate.
  - coord_tot{4}{1} would yield back an array with the x-coordinates of the mesh inside Special Box number 4, ordered from smallest to highest coordinate.

- **vel_tot** [m/s]: Is a cell array variable. Its length is equal to the number of Boxes present in the selected folder. Each Box has a cell of the array assigned.

  In each cell, a new cell array is stored. It has four columns and a number of rows equal to the number of files associated with that Special Box (each file corresponds to a specific time of the simulation). In the first column, the associated time of the row is stored. In columns two to four, the velocity fields u (x-direction), v (y-direction) and w (z-direction) of the mesh inside that Box are stored in that order.

  Each velocity field is a 3-D matrix of the velocity in the associated direction in the center of the square mesh cell determined by the coord_tot variable.

  For example:

  - vel_tot{1}{2,1} would yield back time step number 2 of Special Box 1.
  - vel_tot{3}{15,2} would yield back the x-component of velocity in the center of the mesh cells inside Box 3.
  - vel_tot{4}{3,4}(1,1,1) would yield back the z-component of velocity in the center of the first cell of the mesh inside Box 4.

    In this case, the cell of the mesh corresponds to the one that has the following coordinates:

    * In the x-direction its left and right coordinates are coord_tot{4}{1}(1) and coord_tot{4}{1}(2), respectively.
    * In the y-direction, coord_tot{4}{2}(1) and coord_tot{4}{2}(2).
    * In the z-direction its bottom and top coordinates are coord_tot{4}{3}(1) and coord_tot{4}{3}(2).

Procedure

1. Select manually the *datasb* folder where the files are stored

```
openfold = uigetdir([]);
openfold_seperate = strsplit(openfold,'\');
```

2. Determine the number of files per Special Box present in said folder, taking into account that there are seven special boxes (num_boxes) and the quantity of files in said folder (ubication).

```
ubication = dir([openfold, '\*.dat']);
ind_max = (length(ubication) - 1)/num_boxes;
```

3. ComFLOW program is opened.

```
sf = CMFSnapshot;
```

4. Parallely, for each box the following steps are taken to extract the information from the Special Box files:

   (a) First Special Box file (0) for that box (s_box) is opened.

```
fnm = sprintf('Specialbox%03d_%06d.dat',s_box,0);
dirfnm = fullfile(openfold,fnm);
res = sf.loadFile(dirfnm);
if res~=0
    warning('Unable to load Special Box. End of execution.');
end
```

   (b) Mesh cell coordinates of that Special Box are saved as explained in Output.

```
coord = cell(3,1);
coord{1} = sf.x{1};
coord{2} = sf.y{1};
coord{3} = sf.z{1};
coord_tot{s_box} = coord; %Store coordinates in the cell of
                          %coord_tot associated with s_box.
```

   (c) Velocity fields for that time step are stored.

```
vel{1,1} = sf.var.time;
vel{1,2} = sf.var.u{1};
vel{1,3} = sf.var.v{1};
vel{1,4} = sf.var.w{1};
```

   (d) Analogously, coordinates and velocity fields for the other time steps are obtained:

      i. Corresponding Special Box file is opened.

      ii. Velocity fields for that time step are stored.

5. ComFLOW program is closed.

```
sf.unloadFile();
```

5.1.2 *cal_Ek_parallel* function

As previously stated, this function is used to calculate the kinetic energy at every time step in all the areas of study. To execute these calculations, the function takes as an input the variables output by *read_velocity_parallel*, *coord_tot* and *vel_tot*. Then, returns the results in the form of three new cell array variables named *Ek_tot_t*, *Ek_tot_x* and *Ek_tot_z* corresponding to the kinetic energy produced by the absolute velocity, only the horizontal velocity component, and only the vertical velocity component, respectively.

To carry out these calculations one key assumption is made. Each one of the mesh cells inside the studied area is treated as one particle with its mass concentrated in the center of the said cell, taking the velocity interpolated to this center as the particle's velocity. Then, the kinetic energy contributed for each of these assumed particles is calculated with

$$E_k\,(particle) = \frac{1}{2}\,m_p\,v_p^2 \tag{5.1}$$

$$m_p = \rho\,\Delta x\,\Delta z\,T_h \tag{5.2}$$

where m_p (kg) corresponds to the particle's mass, v_p (m/s) to the particle's velocity, ρ (kg/m^3) to water's density, Δx (m) to the width of the mesh cell, Δz (m) to the height of the mesh cell and T_h to the thickness of the studied area (taken from [27]).

When the mesh cell is only partly inside the studied area, only this part of the mesh cell is considered as a contributing particle to the kinetic energy. Consequently, the mass is considered to be concentrated at the center of this part of the cell, and the corresponding velocity has to be interpolated from the known velocities at the center of the mesh cells. In the left-side picture of Figure 5.4, the known velocities of the special box that contains the studied area are shown in red. Whereas in the right-side picture of the same Figure, the area of study and the different particles that are considered to be part of it are drawn in green. In this last case, the points where the particles are considered to be concentrated are plotted as a point.

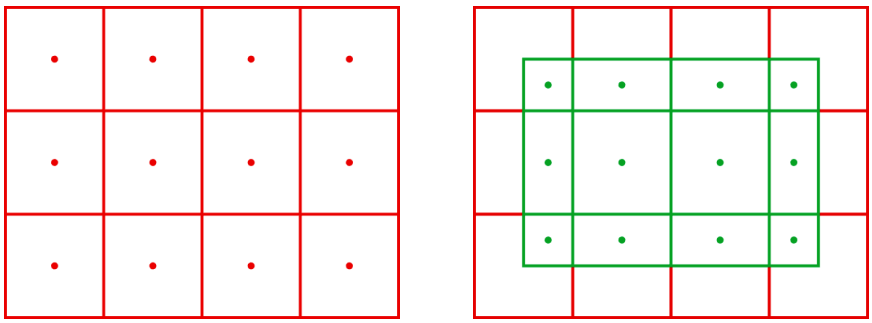

Figure 5.4: Considered Special Box (left) and area of study divided into particles represented in green (right).

Finally, the sum of all the contributing particles of one area of study is the value considered as the kinetic energy contained in that area for one specific time step. The value of this sum is calculated for each time step leading to a series of values that can describe the evolution of the kinetic energy content in one specific area over time.

Input

- **coord_tot** [m]: Same type of variable as the one output by *read_velocity_parallel*. See Section 5.1.1.
- **vel_tot** [m/s]: Same type of variable as the one output by *read_velocity_parallel*. See Section 5.1.1.

Output

- **Ek_tot_t** [J]: Is a cell array variable. Its length is equal to the number of areas of the wave tank studied. Each area has a cell of the array assigned (Cell number one corresponds to area 0, cell number two to area 1, etc.).

  Each of those cells contains a Matrix of 2 columns and a number of rows equal to the number of time steps of the simulation. For each row, in column 1 there is the corresponding time and in column 2 the value of total kinetic energy contained by that area at that time step.

  For example:

  - Ek_tot_t{1}(1,1) would yield back the time of the first time frame for area 0 (Special Box 1).
  - Ek_tot_t{4}(4,2) would yield back the total kinetic energy present in area 3 (Special Box 4) at the 4th time frame analysed.
- **Ek_tot_x** [m/s]: [J]: Its structure is equal to *Ek_tot_t*, the only difference is that instead of containing the information of the absolute kinetic energy in a specific area, stores the X component of said kinetic energy.
- **Ek_tot_z** [m/s]: [J]: Its structure is equal to *Ek_tot_t*, the only difference is that instead of containing the information of the absolute kinetic energy in a specific area, stores the Z component of said kinetic energy.

Procedure

The following steps are taken in parallel for each of the special boxes of which we have coordinates and velocity information stored in *coord_tot* and *vel_tot*.

1. Determine the exact coordinates for that area of study, depending on the special box selected.

```
        if s_box == 1 %Area 0
            xpositions = [2.811,2.920]
        elseif s_box == 2 %Area 1
            xpositions = [3.111,3.220]
        elseif s_box == 3 %Area 2
            xpositions = [3.411,3.520]
        elseif s_box == 4 %Area 3
            xpositions = [3.711,3.820]
        elseif s_box == 5 %Area 4
            xpositions = [4.311,4.420]
        elseif s_box == 6 %Area 5
            xpositions = [4.611,4.720]
        elseif s_box == 7 %Area 6
            xpositions = [4.911,5.020]
        end
```

2. Execute auxiliary Matlab function *Ek_cal*. It calculates the evolution of the kinetic energy content for the selected area.

```
[Ek_tot_t{s_box},Ek_tot_x{s_box},Ek_tot_z{s_box}] = ...
    cal_Ek(coord_tot{s_box},vel_tot{s_box},...
    xpositions(1),xpositions(2),-0.278,-0.0895);
```

The following steps are taken in this auxiliary function.

(a) Determine number of time steps of the simulation,

```
ind_max = length(vtot(:,1));
```

(b) For each time step, go through all mesh cells of the special box, and for each mesh cell follow the following procedure:

i. Determine the size (x_A and z_A) and associated velocities (u_A and w_A) of the assumed particle, depending on where the mesh cell is located in relation to the studied area.
In the following code snippet only two cases are shown, to see all the considered cases see the full code in Appendix B.2.

```
%if the current cell is inside the limits of the Area,
%gather cell size and velocity information
if (x(i)>=xmin) && (x(i+1)<=xmax) && (z(k)>=zmin) &&...
        (z(k+1)<=zmax)
    x_A = x(i+1)-x(i);
    z_A = z(k+1)-z(k);
    u_A = u(i,j,k);
    w_A = w(i,j,k);

%if the Area limit is in the middle of the current cell,
%interpole velocities to the center of the part of the
%mesh cell that is inside the Area limits.
elseif (x(i)<xmin) && (x(i+1)>xmin) && (z(k)>=zmin) && ...
        (z(k+1)<=zmax)
    %Interpole in X. Left side.
    x_A = x(i+1)-xmin;
    z_A = z(k+1)-z(k);
    u_A = itp_lin(mean([x(i+1),x(i)]),mean([x(i+2),x(i+1)
        ]),u(i,j,k),u(i+1,j,k),mean([x(i+1),xmin]));
    w_A = itp_lin(mean([x(i+1),x(i)]),mean([x(i+2),x(i+1)
        ]),w(i,j,k),w(i+1,j,k),mean([x(i+1),xmin]));
```

ii. Calculate the assumed particle's kinetic energy associated with its x-axis and z-axis velocities separately.

```
mass = rho*x_A*Th*z_A;
Ex = Ex + 0.5*mass*u_A.^2;
Ez = Ez + 0.5*mass*w_A.^2;
```

(c) Store the results.

```
Ek_t(ind,1) = t;
Ek_t(ind,2) = Ex + Ez;

Ek_x(ind,1) = t;
Ek_x(ind,2) = Ex;

Ek_z(ind,1) = t;
Ek_z(ind,2) = Ez;
```

5.1.3 *compare_windows* function

This function is used to compare the evolution of the kinetic energy content in the seven studied areas for a different number of cases of study. The function yields back as much plots as different areas of study (in this case seven plots). In each plot, the function draws the kinetic energy evolution in that area for as much cases as desired. Furthermore, for a better insight, the maximum and minimum absolute kinetic energy measured experimentally in [27] are plotted in the form of a dashed line.

Input

- **Ek** : Is a cell array that contains the kinetic energy signals to compare.

  For example:

  - {*Ek_tot_t_m1*,*Ek_tot_t_m2*} would be a cell array containing the output of absolute kinetic energy for two different cases, Mesh 1 and Mesh 2.

- **E_names** : Is a cell array that contains the name of the kinetic energy signals in *Ek* to be plotted in the plot legend.

  Following the example in *Ek*:

  - {'Mesh 1','Mesh 2'} would be a cell array containing the associated names for *Ek_tot_t_m1* and *Ek_tot_t_m2*, respectively.

Procedure

1. Experimental measurements' minimum (*MIN_J*) and maximum (*MAX_J*) values are loaded from the file named *JEN.m*.

```
load('JEN.mat')
```

2. For each area of study, auxiliary function *compare_Ek* is executed.

```
for i=1:7;
    compare_Ek({Ek{1}{i},Ek{2}{i}},E_names,MAX_J(i),MIN_J(i),i)
end
```

   The following steps are taken in this auxiliary function.

   (a) The kinetic energy of that area is plotted for all the introduced cases of study.

```
for i=1:length(E_all)
    plot(E_all{i}(:,1),E_all{i}(:,2),colors(i));
end
```

   (b) The maximum and minimum measured kinetic energy for that area are plotted.

```
plot([0,50],[min_J,min_J],'k--')
plot([0,50],[max_J,max_J],'k--')
```

   (c) The plot legend is drawn.

```
legend([E_name_all,{'Minimum measured','Maximum measured'}])
```

5.2 Water height

Five functions have been developed to carry out the frequency domain analysis of the water height evolution over time of the tank's surface.

On first instance, the function coded in the Matlab file *read_wh.m* (see code in Appendix B.4) is used to read all Snapshot output files stored in one *data* folder and return the information stored in those in the form of one array and one cell array variables named *coord_wh* and *wh_t*, respectively.

On second place, the function coded in the Matlab file *wh_point.m* (see code in Appendix B.5) is used to obtain the evolution of the water height in one exact coordinate of the tank's water surface in the form of an array named *wh_p*.

Thirdly, the function coded in the Matlab file *wavelet_wt.m* (see the code in Appendix B.6) is used to plot the wavelet transform of the water height evolution over time in one point of the tank's surface or, in other words, the array *wh_p* output by *wh_point*.

Next in order, the function coded in the Matlab file *fft_wt.m* (see the code in Appendix B.6) is used to plot the fast Fourier transform of the water height evolution over time in one point of the tank's surface. Or, in other words, the array *wh_p* output by *wh_point*.

Finally, the function coded in the Matlab file *harmonic_analysis_wt.m* (see the code in Appendix B.8) is used to plot the evolution over space of the magnitude of the first and second harmonic signals decomposed from the surface's water height. Consequently, this function takes as an input the evolution over time of the surface's water height of all the points along the wave tank or, in other words, the cell array and array output by *read_wh*, *coord_wh* and *wh_t*.

In the following pages, all these functions will be further explained. In the case of the last three functions, they yield back three different types of plots that will be further explained in Chapter 7. In Figure 5.5 a diagram block of the water height analysis procedure is shown.

5.2.1 *read_wh* function

As previously stated, this function is used to read all Snapshot output files saved in one *data* folder and return the information stored in those in the form of one array and one cell array variables named *coord_wh* and *wh_t*, respectively.

Snapshot files save information of the whole domain of the simulation. The quantity of Snapshots taken is configured in *comflow.in* file and are proportionally spread over time. Each Snapshot has its own file. All these files contain information about the coordinates of which the surface's water height are taken and its corresponding values.

Output

- **coord_wh** [m]: Is an array. Corresponds to the x-axis coordinates of which we know the evolution of the water height. This points are located in the center of each one of the mesh cells in the water surface. The more mesh cells, the longer is *coord_wh*.

- **wh_t** [m]: Is a cell array of 2 columns and as many rows as Snapshot files in the selected folder. Each Snapshot file corresponds to one specific time of the simulation. For each row, on the first column time is stored while on column number 2 there is an array with the water height at the coordinates stated in *coord_wh*.

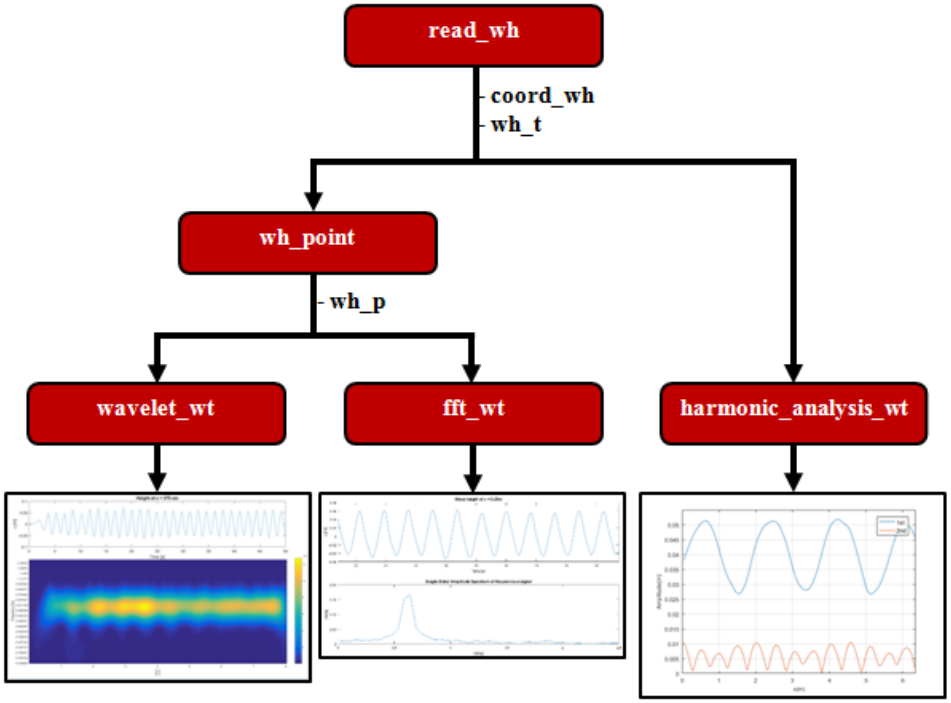

Figure 5.5: Water height analysis procedure for a single simulation case (one mesh).

For example:

- wh_t{1,1} yields back the row's time.
- wh_t{34,2}(3) yields back the water height at coord_wh(3) at row 34's associated time.

Procedure

1. Select manually the *data* folder where the files are stored.

```
openfold = uigetdir([]);
openfold_seperate = strsplit(openfold,'\');
```

2. Determine the number of Snapshot files in said folder.

```
ubication = dir([openfold, '\*.dat']);
ind_max = ((length(ubication) - 2));
```

3. ComFLOW program is opened.

```
sf = CMFSnapshot;
```

4. Save the coordinates of the points along the tank from where the water height has been saved in the snapshot files.

```
wh = sf.waterHeight(); %Auxiliar variable
coord_wh = wh.xc;
```

5. For each of the Snapshot files stored in the selected folder the following steps are taken.

   (a) Load Snapshot file.

```
fnm = sprintf('cmf%1dd%04d.dat',3,ind);
dirfnm = fullfile(openfold,fnm);
res = sf.loadFile(dirfnm);
if res~=0
    warning('Unable to load snapshot. End of execution.');
    return
end
```

   (b) Save the water height along the tank for the time step associated with the file.

```
wh = sf.waterHeight(); %Auxiliar variable
wh_t{ind+1,1} = sf.var.time;
wh_t{ind+1,2} = wh.wh;
```

6. ComFLOW program is closed.

```
sf.unloadFile();
```

5.2.2 *wh_point* function

As previously stated, this function is used to obtain the evolution of the water height in one exact coordinate of the tank's water surface, in the form of an array named *wh_p*. To do so, the function takes as an input the variables output by *read_wh*, *coord_wh* and *wh_t*, and interpolates this information to the desired coordinate, x.

Input

- ***x*** [m]: Is the x-axis coordinate of which the evolution of the water height is wanted.
- **coord_wh** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.1.
- **wh_t** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.1.

Output

- **wh_p** [m]: Is a cell array of 2 columns and as many rows as Snapshot files in the selected folder. It corresponds to the water height evolution at one point of the wave tank. For each row, on the first column time is stored while on column number 2 the water height associated to that time is stored.

Procedure

1. For each time step of which wave height information is known the following step is taken.

   (a) Water height is interpolated in the x coordinate input and saved.

```
wh_p(i_t,2) = interp1(coord_wh,wh_t{i_t,2},x);
wh_p(i_t,1) = wh_t{i_t,1};
```

5.2.3 *wavelet_wt* function

As previously stated, this function is used to plot the wavelet transform of the signal of the evolution of the water height in one exact coordinate of the tank's water surface. Consequently, the function takes as an input the variable output by *wh_point*, *wh_p*. Moreover, is necessary to input the expected period of said signal in s, *T*.

This code is based on a code developed by Ocean Grazer's collaborator Yanji Wei.

To understand what is a wavelet transform plot and how to interpret it, see Chapter 7.

Input

- **wh_p** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.2.
- **T** [s] : Is the theoretical period of the *wh_p* signal. In the case of this thesis, this period is 1.62 s.

Procedure

1. Ensure that the input data has a value of the water height for each 0.01 s.

```
t_i = 0:0.01:49.5;
wh_i = interp1(wh_p(:,1),wh_p(:,2),t_i);
len_wh_i = length(wh_i);
```

2. Determine the scale range of the transform based on the desired frequency spectrum and wavelet function.

```
Fs = 100; %Frequency of the signal (1/(0,01 s)).
fc = centfrq('cmor1-1'); %Center frequency of the wavelet function 'cmor1
    -1'.
freqrange = [0.5/T 5/T]; %Selected frequency range to be plotted.
scalerange = fc./(freqrange*(1/Fs)); %Scale range to be plotted (
    Transformed from freqrange).
```

3. Execute the wavelet transform.

```
scales = linspace(scalerange(1),scalerange(end),100);
Coeffs = cwt(wh_i(1:end-1),scales,'cmor1-1');
pfreq = scal2frq(scales,'cmor1-1',1/Fs);
periods = pfreq;
```

4. Plot the wavelet transform.

5.2.4 *fft_wt* function

As previously stated, this function is used to plot the fast fourier transform of the water height evolution over time in one point of the tank's surface or, in other words, the array *wh_p* output by *wh_point*.

To understand what is a fast fourier transform plot and how to interpret it, see Chapter 7.

Input

- **wh_p** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.2.

Procedure

1. Ensure that the input data has a value of the water height for each 0.01 s and discard the first seconds of signal because of its unsteady nature.

```
t_i = uT*5:0.01:49.5;
wh_i = interp1(wh_p(:,1),wh_p(:,2),t_i);
len_wh_i = length(wh_i);
```

2. Execute the fast fourier transform.

```
transform = fft(wh_i);
```

3. Obtain the single-sided amplitude spectrum as well as the frequency spectrum.

```
Fs = 100; %Frequency of the signal (1/(0,01 s)).
P2 = abs(transform/len_wh_i);
P1 = P2(1:len_wh_i/2+1);
P1(2:end-1) = 2*P1(2:end-1); %Single-sided amplitude spectrum.
f = Fs*(0:(len_wh_i/2))/len_wh_i; %Frequency domain.
```

4. Plot the fast fourier transform.

5.2.5 *harmonic_analysis_wt* function

As previously stated, this function is used to plot the evolution over space of the magnitude of the first and second harmonic signals decomposed from the surface's water height. This harmonic signals are the result of the harmonic analysis of the input signal.

This code is based on a code developed by Ocean Grazer's collaborator Yanji Wei.

To understand what is an harmonic analysis and how to interpret its plot, see chapter 7.

Input

- **coord_wh** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.1.
- **wh_t** [m]: Same type of variable as the one output by *read_wh*. See Section 5.2.1.

Procedure

1. Convert the input information into a more suitable Matlab structure by executing auxiliary function *NumericalWaveData*. Furthermore, only the data of the water height in the coordinates in the range that goes from 0.05 m to 6.35 m are considered.

```
[time_N,data_N] =NumericalWaveData(coord_wh,wh_t);
x_N=coord_wh(find(coord_wh>0.05,1):find(coord_wh>6.35,1));
```

2. Decompose the signal into first and second order harmonic signals, by estimating its corresponding amplitude for each of the points along the tank of which we have data.

```
m=length(time_N);
for ii=1:length(x_N);
    for i=1:m;
        for j=1:n;
            A_N(i,2*j-1)=cos(j*w*time_N(i));
            A_N(i,2*j)=sin(j*w*time_N(i));
        end
    end
    coef_N(:,ii)=A_N\data_N(:,ii);
    data_a_N=A_N*coef_N(:,ii);
    data_a_N_all(:,ii)=data_a_N;
end
for ii=1:length(x_N);
    for j=1:n
       amp_N(j,ii)=sqrt(coef_N(2*j-1,ii)*coef_N(2*j-1,ii)+...
                        coef_N(2*j,ii)*coef_N(2*j,ii));
    end
end
```

3. Plot the amplitude of both harmonic signals along the considerate coordinate range.

```
figure(8);
```

Chapter 6

Time domain analysis of results

This chapter presents and analyses the kinetic energy content calculated in each of the studied areas for the different CFD simulations executed (different mesh refinement). Furthermore, a comparison with the corresponding experimental measurement taken in Jenifer Brenes' thesis [27] is carried out for each area, as the most direct way of validation of said measurements. More precisely, these results have been compared in the following two ways.

On the one hand, the evolution over time of the kinetic energy content is analysed and compared. Firstly, a comparison of the simulations' results is made for each separate area of study to try to validate the convergence of the simulation. This comparison is carried out for the full fifty simulated seconds to see if the simulation results converge during all the simulation. In the second place, these results are compared with the maximum and minimum measured values by Jenifer, to get an idea of the deviation between the experimental measurements and the simulation results, and evaluate if it is possible to validate the previous.

On the other hand, a comparison of the mean over time of the kinetic energy content in each area is made. The different studied simulation cases and the experimental measurements are included in this comparison as well. In the case of the simulations' results, only the last twenty seconds of the simulation have been taken into account, when the simulation is already in the steady state.

As a reminder, and as explained in chapter 4, there are a total of seven studied areas of which results are obtained and compared. These are distributed along the tank as indicated in Figure 4.6, being area 0 the closest to the wave paddle and area 6 the closest to the Beach. Regarding its size, these areas have a width of 10.90 cm and a height of 18.85 cm.

6.1 Kinetic energy content evolution

In a first instance, a comparison of the kinetic energy evolution over time of the differently meshed simulations have been carried out. These comparisons have been made for each of the seven areas of study.

As commented, this comparison aims to corroborate the convergence of the CFD simulation solution regarding the spatial discretisation. This is because the changes in the solution between meshes become smaller as the mesh is more refined following an asymptotic behaviour. And, consequently, when refining the mesh has no effect on the simulation solution, the CFD simulation is considered converged.

In this first analysis, only the comparison of results in area 0 is shown, as the same conclusions can be obtained by analysing the other areas. Figures 6.1, 6.2, 6.3 and 6.4 show the comparison of kinetic energy evolution in area 0 between Meshes 1 and 2, 2 and 3, 3 and 4, and 4 and 5, respectively. This figures as well as the ones corresponding to areas 1 through 6 can be seen in Appendix C.

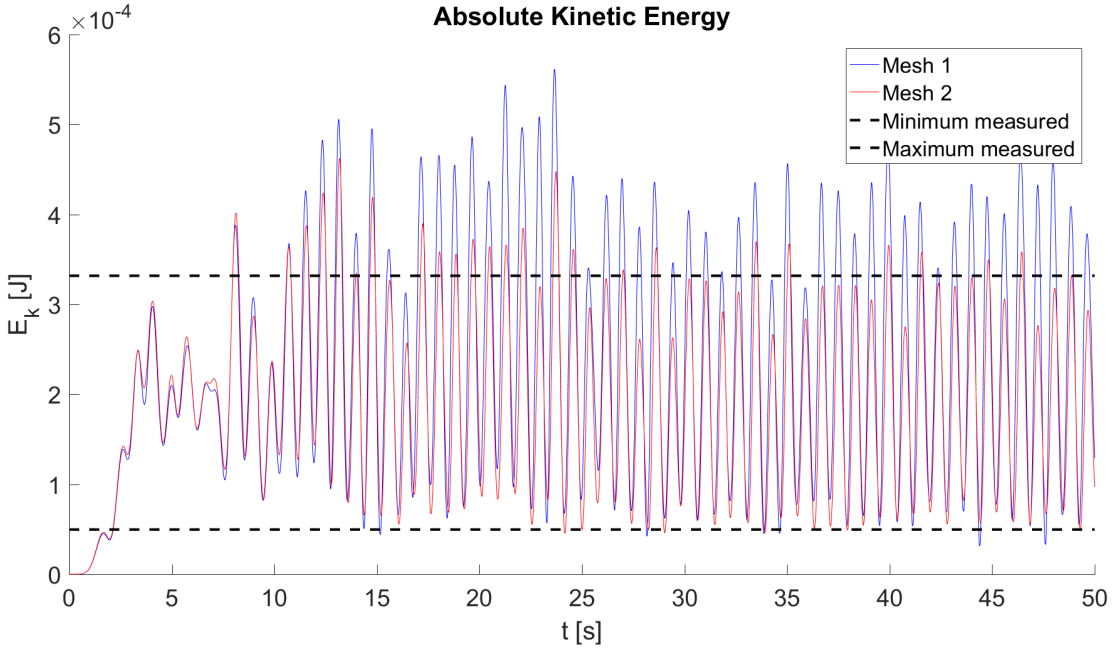

Figure 6.1: Kinetic energy content evolution in area 0 of Mesh 1 and Mesh 2.

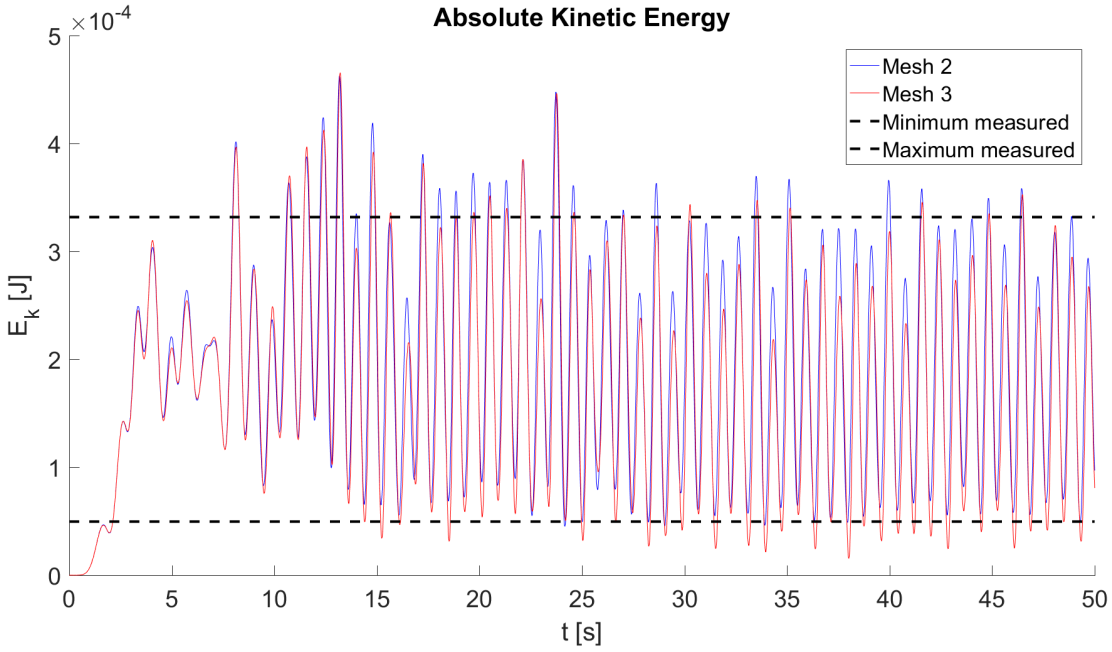

Figure 6.2: Kinetic energy content evolution in area 0 of Mesh 2 and Mesh 3.

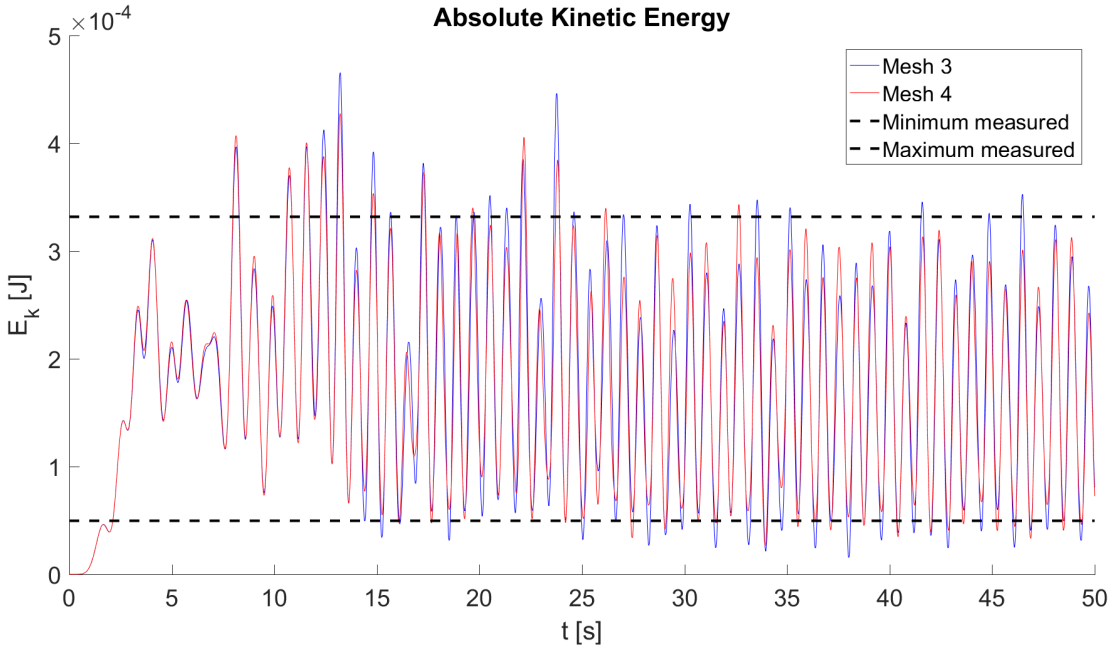

Figure 6.3: Kinetic energy content evolution in area 0 of Mesh 3 and Mesh 4.

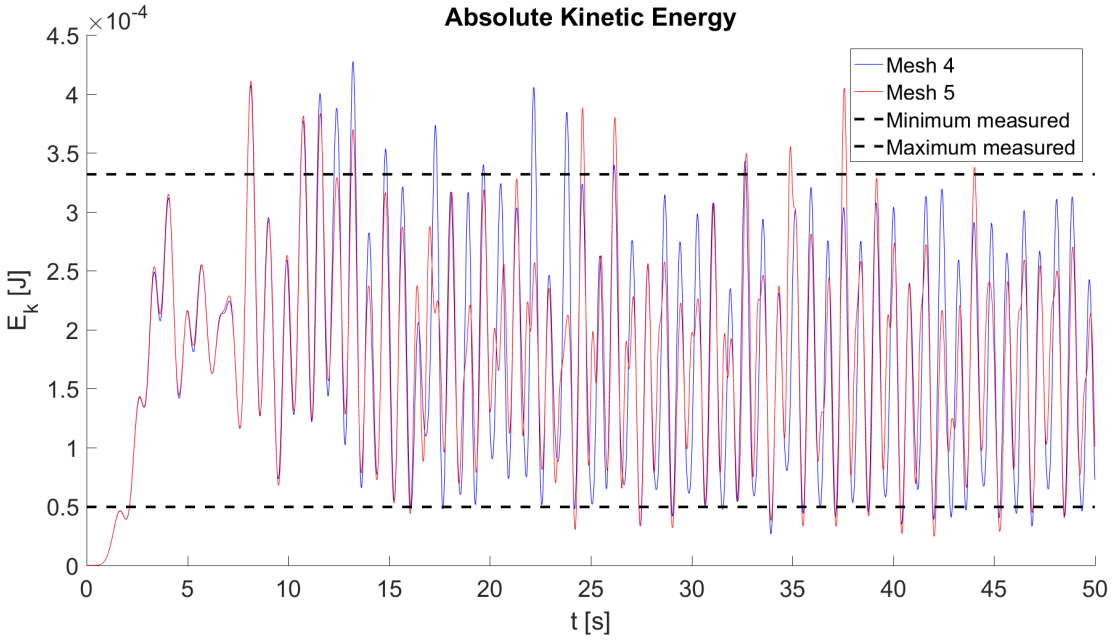

Figure 6.4: Kinetic energy content evolution in area 0 of Mesh 4 and Mesh 5.

The clearest conclusion from analysing the previous figures is the fact that, as the mesh is more refined, the results do not converge to one specific solution. The values of the kinetic energy at different times don't match between meshes, even between the two more refined meshes, even though in these cases the size of the various mesh cells is really small and the simulations take days (around five and seven days for Mesh 4 and 5, respectively).

Secondly, the comparison with the experimental results can also be made in these figures. The dashed lines present in each figure represent the maximum and minimum measured kinetic energy content for that distinct area. If we take a look at Figure 6.4, is possible to see that they concur approximately with both meshes' results.

This does not happen in all the studied areas. Figures 6.5, 6.6, 6.7, 6.8, 6.9 and 6.10 correspond to the comparison between the kinetic energy signals of Mesh 4 and 5 for areas 1, 2, 3, 4, 5 and 6, respectively. Taking a look to all these figures is possible to see that the maximum and minimum measured values of the kinetic energy content don't match with the simulation results in all the areas of study. Sometimes the measured values are lower and sometimes higher.

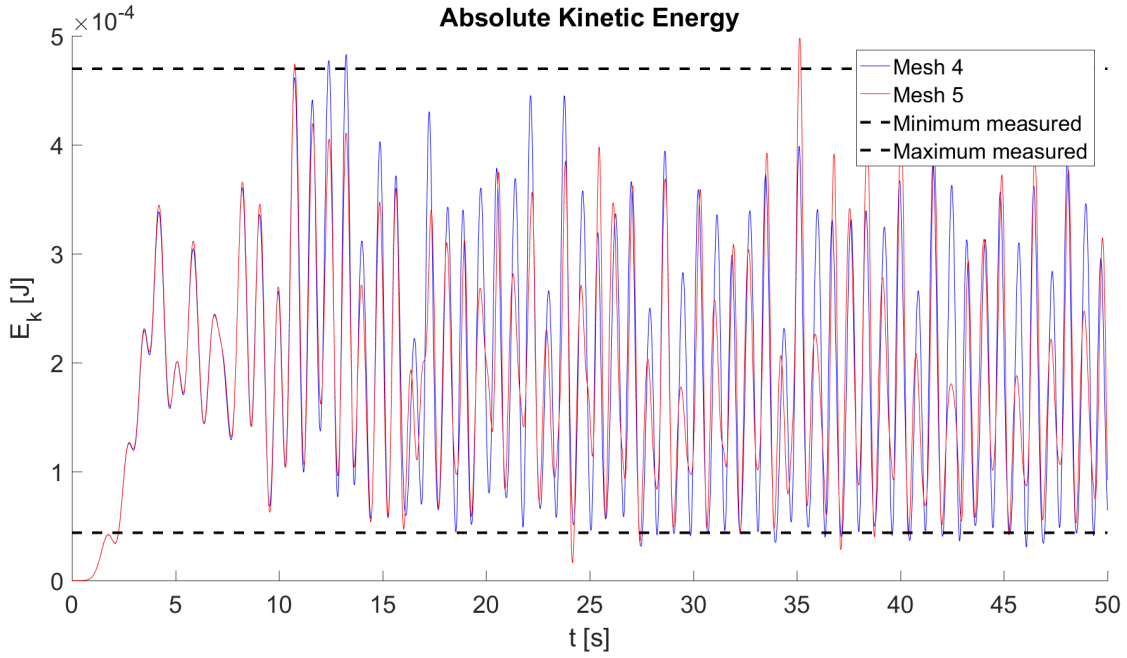

Figure 6.5: Kinetic energy content evolution in area 1 of Mesh 4 and Mesh 5.

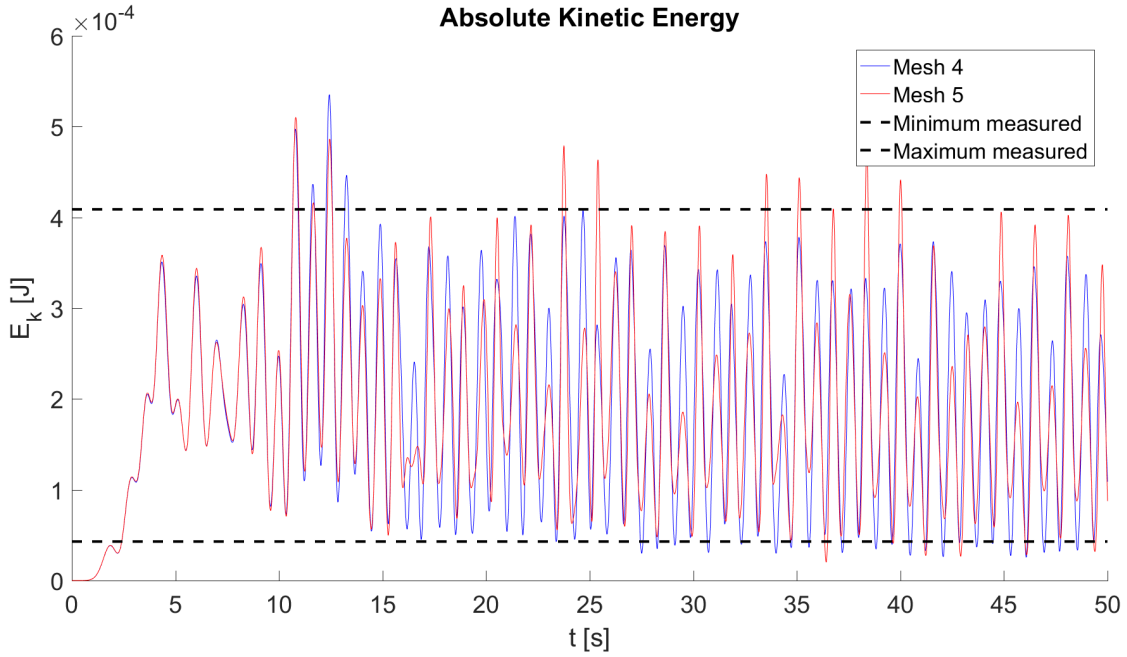

Figure 6.6: Kinetic energy content evolution in area 2 of Mesh 4 and Mesh 5.

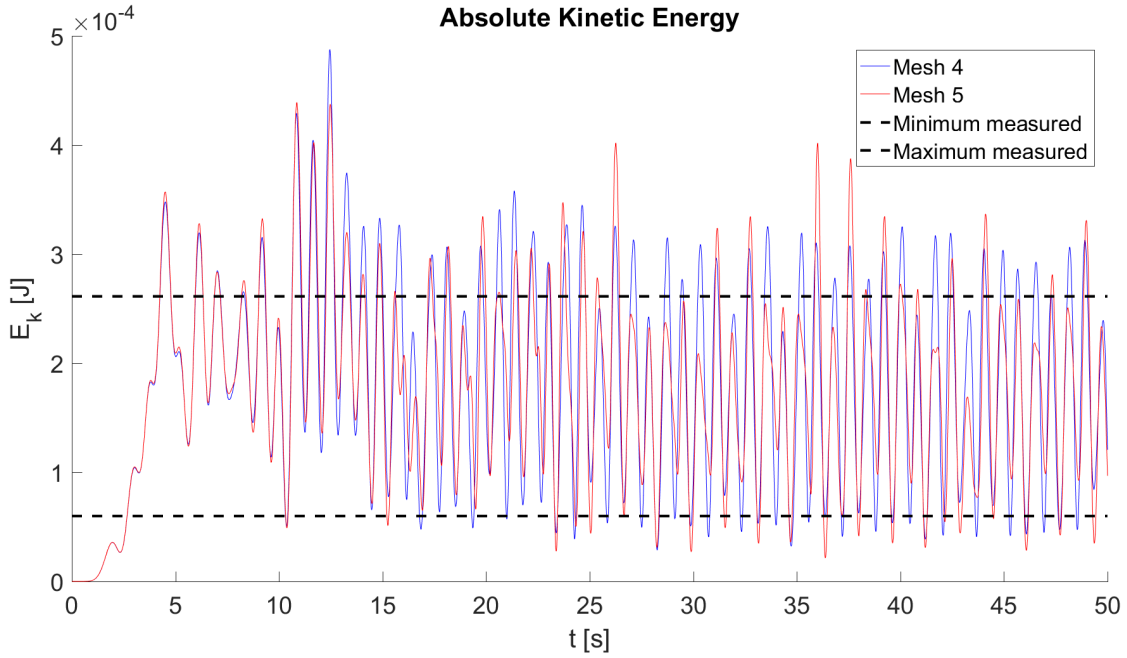

Figure 6.7: Kinetic energy content evolution in area 3 of Mesh 4 and Mesh 5.

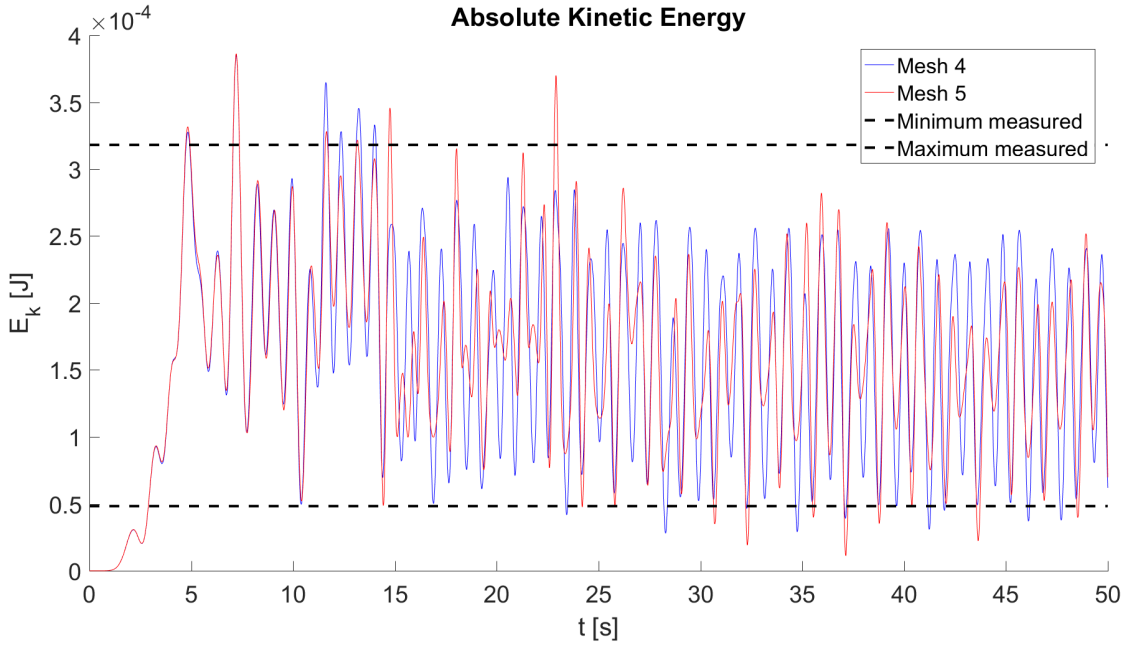

Figure 6.8: Kinetic energy content evolution in area 4 of Mesh 4 and Mesh 5.

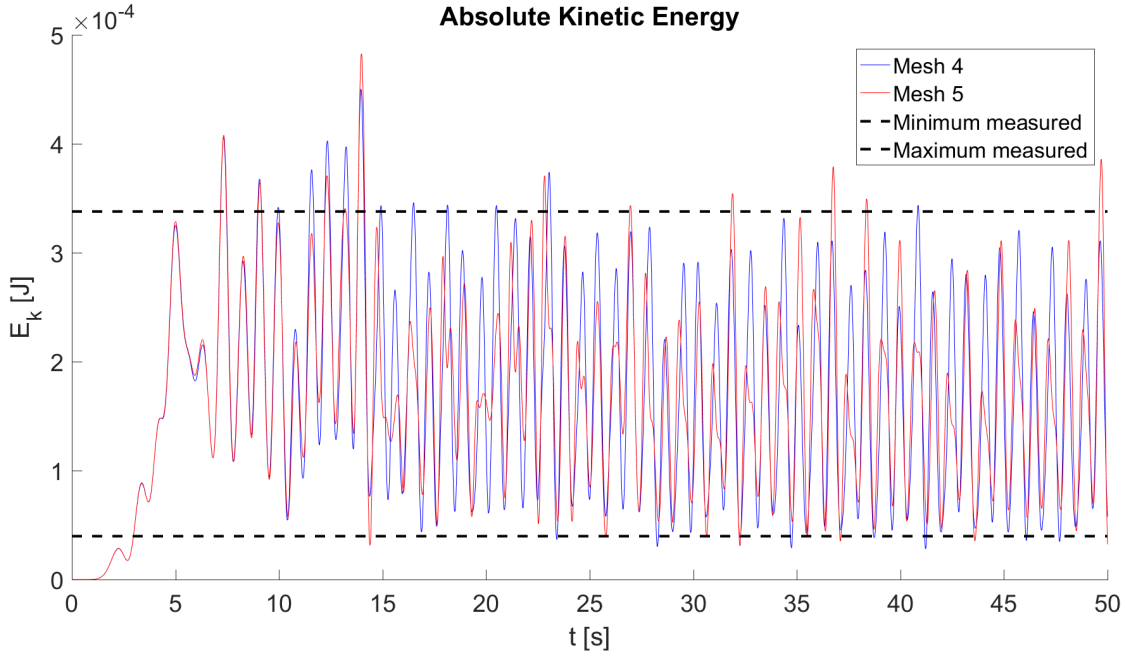

Figure 6.9: Kinetic energy content evolution in area 5 of Mesh 4 and Mesh 5.

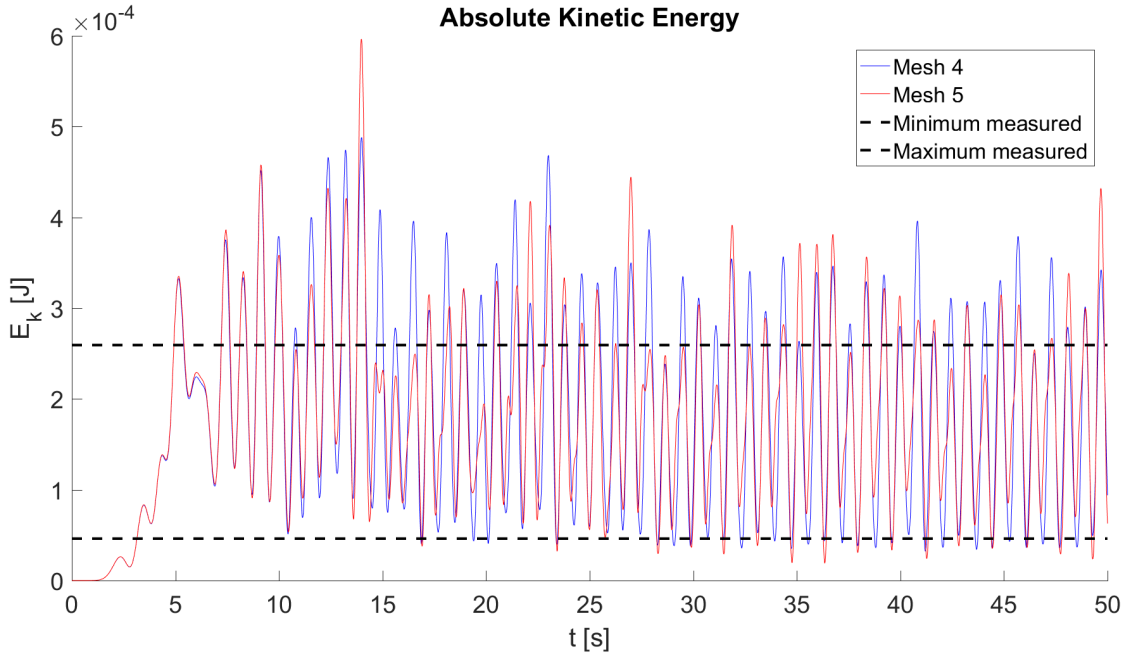

Figure 6.10: Kinetic energy content evolution in area 6 of Mesh 4 and Mesh 5.

Furthermore, theoretically, the kinetic energy evolution signal should follow a pseudo-sinusoidal progression. Taking a look at the kinetic energy evolution in all the plots with the two more refined meshes, various disturbances of these signals can be perceived. These disturbances are more clear in the case of *Mesh 5* and could indicate the presence of reflection of the waves in the wave tank, which may be the reason behind the non-convergence of the simulation.

To confirm and analyse the presence of this and other physical phenomena, a frequency analysis of the water height of the tank's surface was carried out which it is shown in Chapter 7. Moreover, another important reason for which this frequency analysis was decided to be executed, is to try to explain the reason behind the non-convergence of the simulation solution for the five different meshes studied.

6.2 Average Kinetic energy content

In this section, the average kinetic energy content calculated in each one of the areas for the different meshes are presented. Furthermore, the average kinetic energy experimentally measured for each one of the areas are also shown. The main reason why the data is displayed this way is, on the one hand, to visualise how the energy content varies depending on the refinement of the mesh; and, on the other hand, because is another way to compare and validate the taken experimental measurements.

In the case of the simulation results, only the last twenty seconds of simulation (when the steady state had been already reached) were considered to calculate said average.

In Figure 6.11 said information is shown.

Firstly, if we compare the results for the different meshes, even though as the mesh is more refined the difference between the averages is smaller, the convergence of the simulation is again non-observed.

On second place, comparing the averages of the more refined mesh results (Mesh 4 and Mesh 5) with the ones of the experimental measurements, the pattern observed in Section 6.1 is shown again. Depending on the area, the averages of the simulation are closer or further of the ones of the experimental measurements.

Despite this, the evolution of the averages in Mesh 5 and the experimental measurements have a certain resemblance. Between area 0 and area 1, the averages increase; between area 1 and 2, decrease; between area 2 and 3 the decrease is more pronounced; in area 4, the values are again similar; and, finally, increase again in area 5.

Consequently, no definitive conclusion can be reached regarding the convergence of the solution nor the validation of the experimental measurements. Therefore, a frequency analysis of the water height of the tank's surface was carried out in Chapter 7 to investigate the reasons on the non-convergence of the simulation.

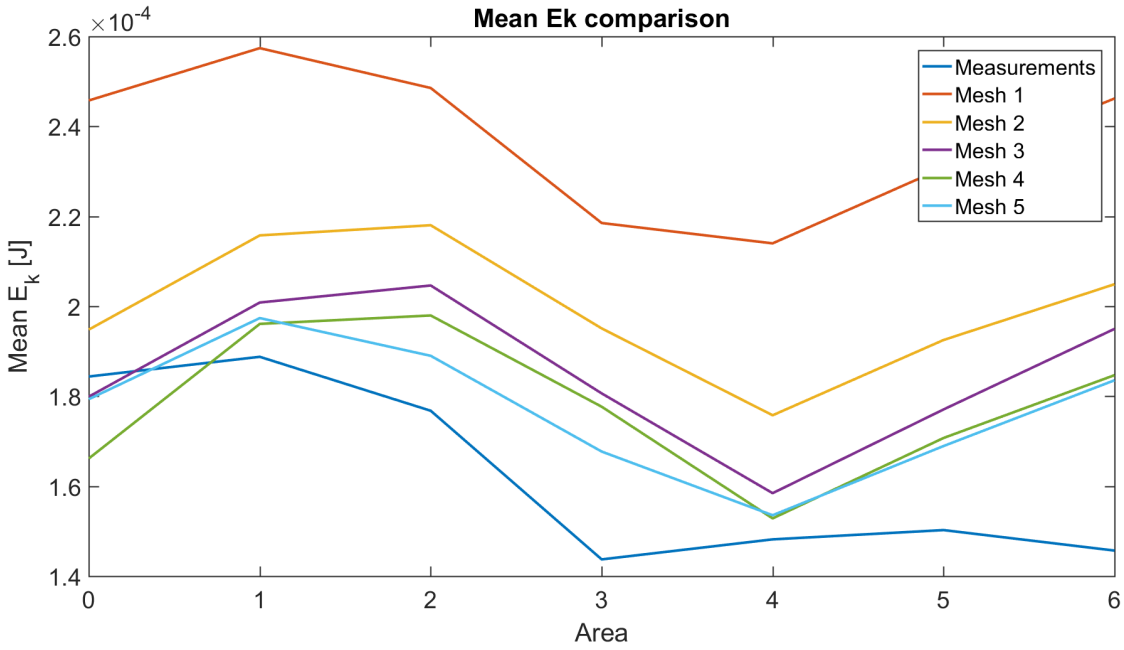

Figure 6.11: Kinetic energy content evolution in area 6 of Mesh 4 and Mesh 5.

Chapter 7

Frequency domain analysis of results

This chapter presents the frequency domain analysis of the water height evolution of the wave tank's water surface for all the simulated cases.

This analysis is carried out to:

- Confirm the suspicion that a great amount of wave reflection is happening in the CFD simulations (see Chapter 6).
- Study the existence of other possible physical phenomena.
- Find an explanation on why the simulation results do not converge with the refinement of the mesh.

This frequency analysis has been done in three steps, using a different tool in each one. First, a continuous wavelet transform analysis of the signal has been done in Section 7.1. On second place, a Fast Fourier transform analysis of the signals has been carried out in Section 7.2. Finally, in Section 7.3, an Harmonic analysis of said signals has been executed.

In the case of the continuous wavelet and fast Fourier transform analysis, the wave height in five different points of the wave tank's water surface have been analysed. Being the simulations executed two-dimensional, only the x-coordinate is necessary to determine the exact position of these points. They are separated from one another by 0.75 m, with the central point being almost in the center of the central area of study. Consequently, the coordinates of these points are the ones shown in Table 7.1 and represented in Figure 7.1.

Table 7.1: X-Coordinates of the water tank points from which the water height has been analysed in the Continuous Wavelet and Fast Fourier transform analysis.

	Point 1	Point 2	Point 3	Point 4	Point 5
X [m]	2.25	3.00	3.75	4.50	5.25

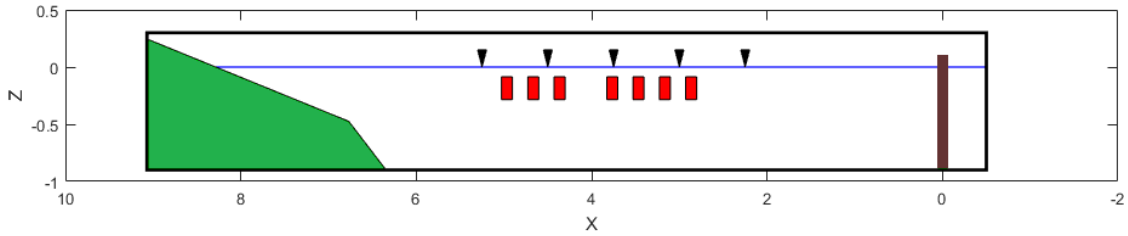

Figure 7.1: Points of the wave tank of which its water surface height has been analysed in the Wavelet and Fast Fourier transform analysis (black triangles).

On the contrary, in the case of the Harmonic analysis, the wave height of the wave tank's water surface of the points between the coordinates 0 m (where the wave paddle is located) and 6.35 m (where the beach starts) have been analysed.

7.1 Continuous Wavelet transform

7.1.1 Introduction

A wavelet is a function that has a wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero, which its mean is equal to zero. Wavelets are the base of the wavelet transform, which "cuts up data or functions or operators into different frequency components" [35].

Each continuous wavelet transform has what is known as the mother wavelet, which is a basic function denoted as $\psi(t)$ whose evolution over time starts at zero, has a certain evolution, and goes back to zero. In the case presented in this chapter, the mother wavelet is a complex morlet wavelet like the one represented graphically in Figure 7.2.

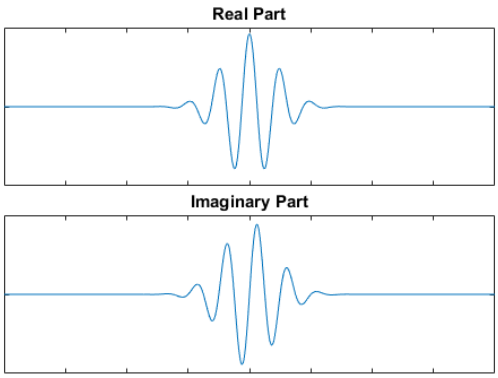

Figure 7.2: Complex Morlet wavelet.

This mother wavelet is scaled and translated forming a basis set of wavelet functions. This set of functions have the same tendency as the mother wavelet, but with different periods and instants in which its output is a number different to zero. This set is denoted by

$$\left\{ \psi_{s,u}(t) = \frac{1}{\sqrt{s}} \psi \frac{t-u}{s} \right\} \Bigg|_{u \in \mathbb{R}, s \in \mathbb{R}^+} \tag{7.1}$$

where u is the translating (time) parameter, indicating which region we concern and s is the scaling (frequency) parameter [36].

Then, this set of wavelets are used to decompose the signal in the continuous wavelet transform. Depending on the wavelets used to describe the studied signal, we would know which frequencies have more or less weight at each time in the studied signal. In consequence, the continuous wavelet transform possesses the ability to analyse the temporal evolution of the frequency content of a given signal [37].

Normally, the wavelet transform is represented in a color plot where the two axis correspond to frequency (Hz) and time (s), and the intensity of the color depends on the percentage of influence of a frequency in the transformed signal at a specific time. In the case presented in this Section, in the x-axis time has been substituted by

$$t \cdot \frac{c}{2L} \tag{7.2}$$

where $c = \frac{\lambda}{T}$ (m/s) is the celerity in which waves travel in the simulation, L (m) is the distance between the wave paddle and the wave tank's beach, T (s) corresponds to the simulated wave period (see 'High Wave' profile in Table 4.1) and λ (m) is the wavelength calculated with the linear wave theory's dispersion relationship (see (2.20)).

In other words, in the following wavelet plots, the x-axis corresponds to the number of times that a single wave of the simulated characteristics can go from one specific point of the tank to the beach, back to the wave paddle and, finally, arriving at the same specific point again because of reflection.

7.1.2 Analysis

In a first instance, a comparison between the wavelet transform plots of the tank's water surface height has been carried out at each of the five points commented previously in Table 7.1. At each point, the signals of the five different simulated meshes have been compared. Figures 7.3, 7.4, 7.5, 7.6 and 7.7 show the wavelet transform plot of the water surface height at $x = 3.75$ m for Mesh 1, 2, 3, 4 and 5, respectively. This Figures, as well as the ones corresponding to the other points of Table 7.1, can be seen in Appendix D.

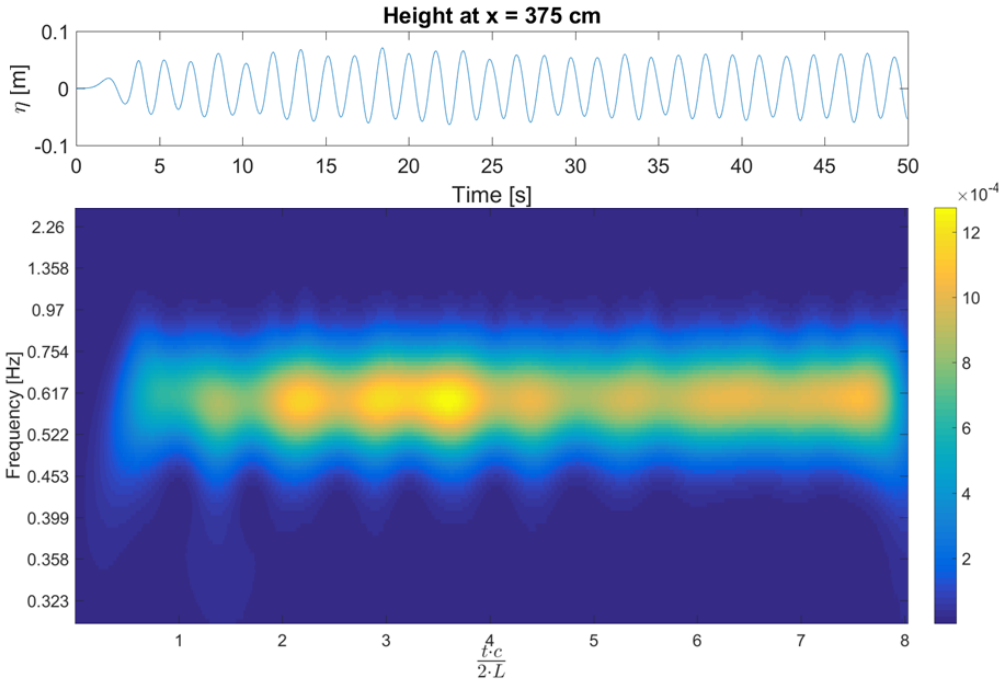

Figure 7.3: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 1.

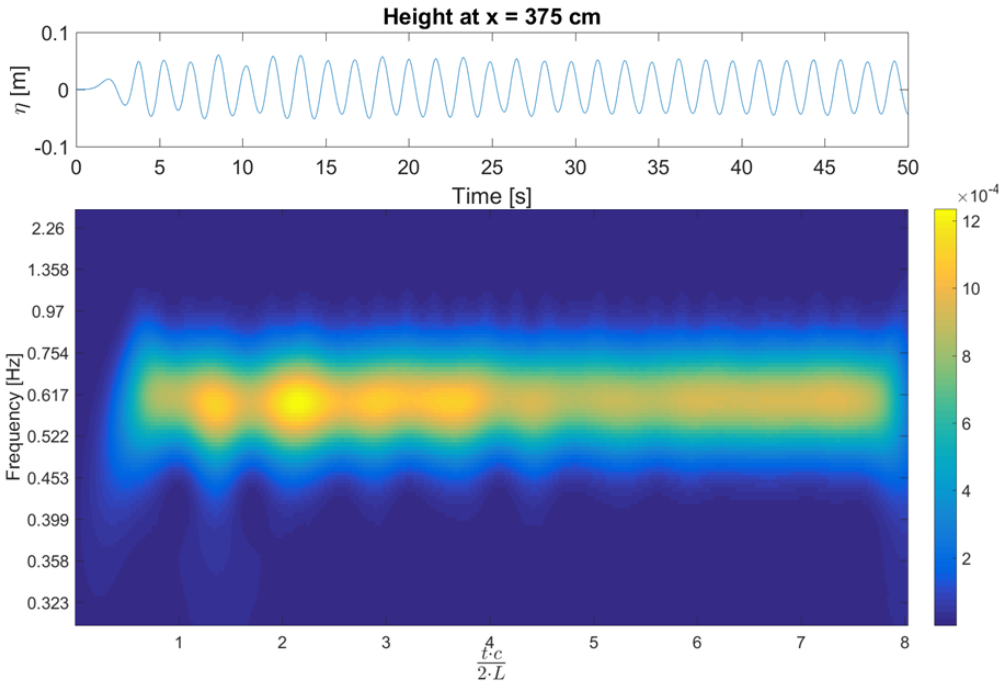

Figure 7.4: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 2.

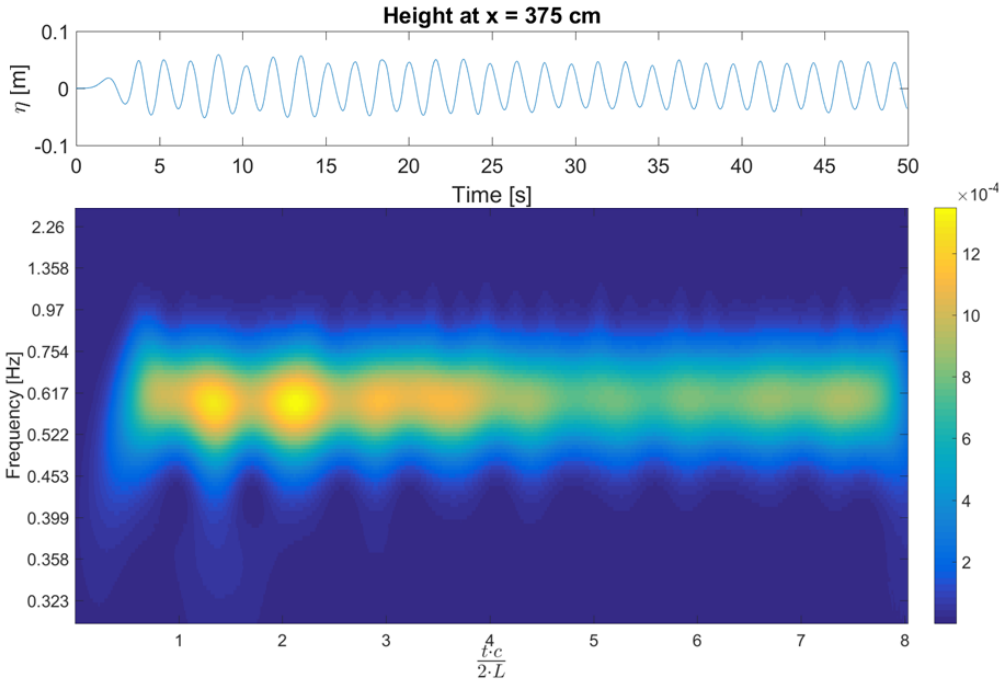

Figure 7.5: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 3.

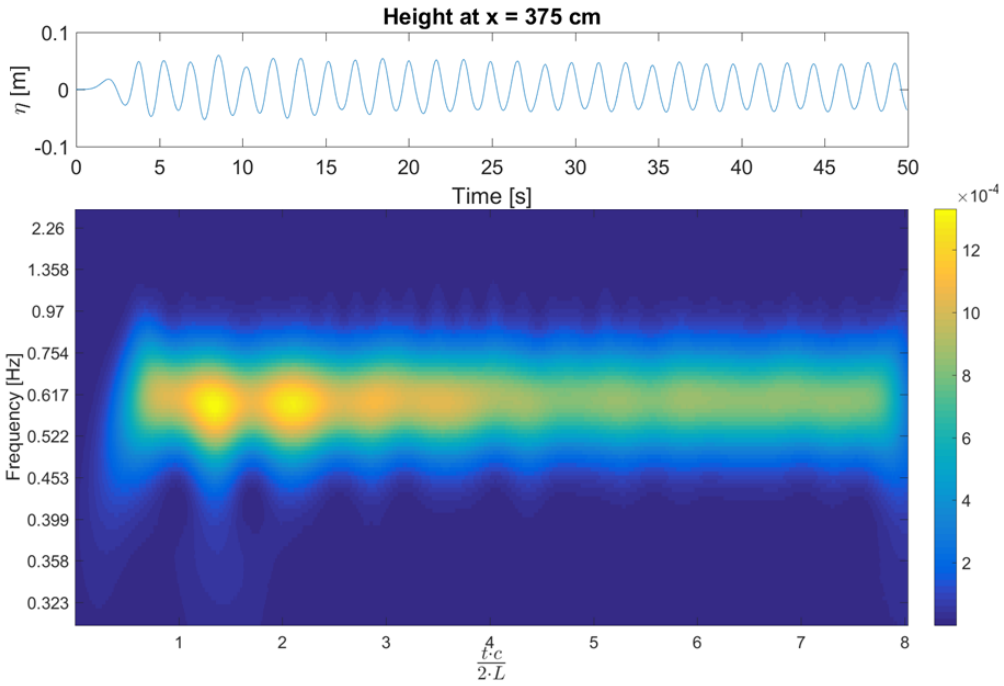

Figure 7.6: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 4.

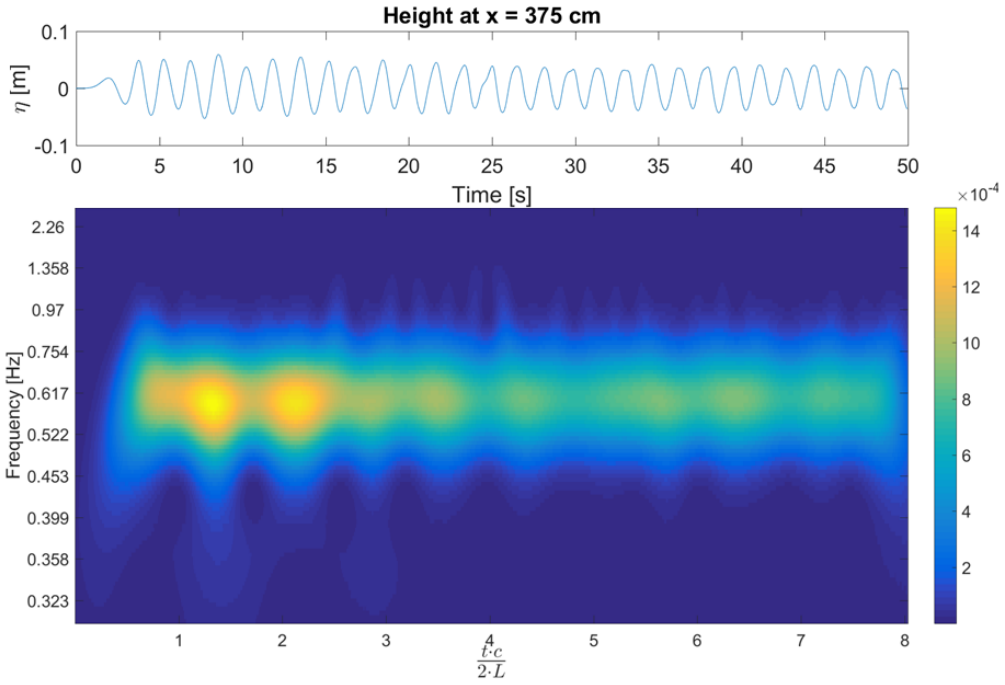

Figure 7.7: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 5.

Taking a look at all these figures, a very similar behaviour of the different signals can be detected, specially in the first seconds of simulation for Meshes 2, 3, 4 and 5. In these cases, at 1.2 and 2.2 x-axis units there are two high-intensity (yellow) spots. Which means that, for all these signals, at 1.2 and 2.2 the frequency components that have more influence on said signals are the ones more closer to the configured wave's frequency ($f_w = 0.617\,Hz$). This indicates that the water height behaviour does not vary much between meshes, specially in these first seconds of simulation.

Between these two points there is a clear discontinuity of the high-intensity area. This fluctuation of the wavelet plot is produced by the wave reflection inside the tank. In fact, the expected behaviour of the wavelet plot in case of reflection is that this pattern is repeated over time: A high-intensity spot followed by a discontinuity repeated on and on and on. The distance between this discontinuities is of approximately one x-axis unit, which is the time that a wave has to travel to both tank's ends and go back to the same point. The former means that when a wave comes back to the same point, the same cycle happens again.

Although this pattern is more clear in the first seconds of simulation, it can be also detected in the rest of the simulation with high-intensity spots of less intensity. This can be seen specially in the case of the wave height at $x = 5.25$ m in Figure 7.8.

Moreover, still analysing the wavelet plot in Figure 7.8, there can be seen that in a lot of the x-axis coordinates a widening of the frequency spectrum happens, forming what seems like a light-blue line. This widening of the spectrum is another effect produced by wave reflection.

In all the previous figures, this commented pattern is more difficult to perceive in the last few seconds of simulation. The commented high-spots have less intensity as the end of the simulation approaches, and the differences with the discontinuities' intensity are almost 0. This could mean that, regarding the wave reflections, steady state has been reached. There is probably a big amount of reflected waves that creates this almost uniformity of the signal. To gain insight on which are the

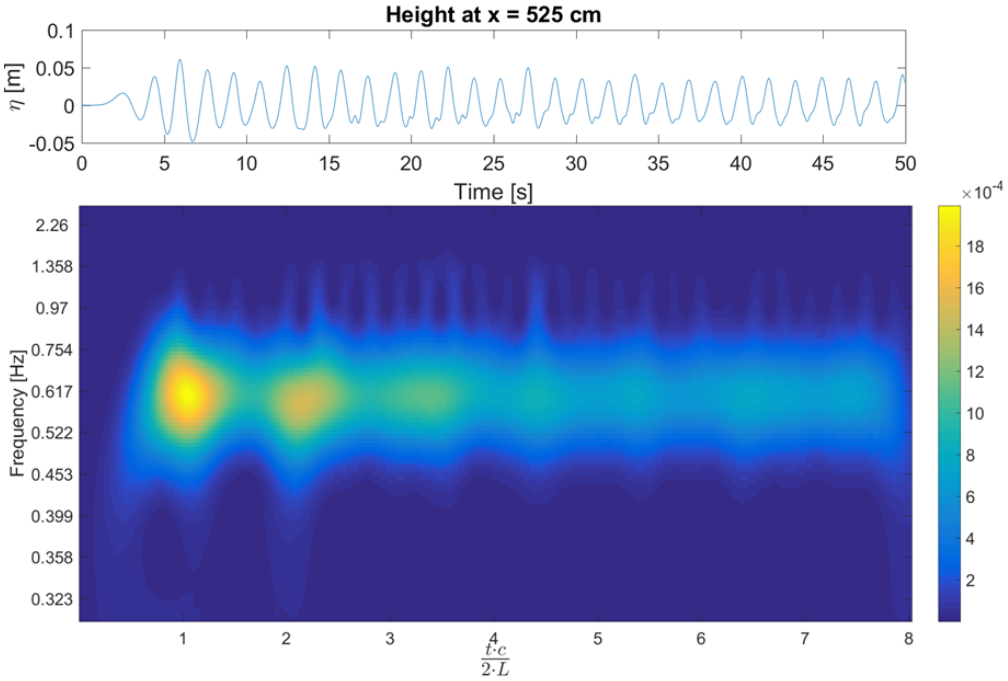

Figure 7.8: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 5.

frequency components present in these last few seconds, a Fast Fourier transform and Harmonic analysis of the same signals have been done. More concretely, starting at x-coordinate 5 (more or less, at the 30.8 s spot).

7.2 Fast Fourier transform

7.2.1 Introduction

The Fast Fourier transform is an algorithm that is used to obtain the Discrete Fourier transform of a specific signal. Fourier analysis converts a signal from its original domain (time in the present case) to a representation in the frequency domain and vice versa. The former is accomplished by breaking down the original signal into a series of sinusoidal terms, each with a unique magnitude, frequency, and phase.

Given a time signal, the expression that defines its Discrete Fourier transform is

$$F_n \equiv \sum_{k=0}^{N-1} f_k \, e^{-i2\pi kn/N} \qquad n = 0, ..., N-1 \tag{7.3}$$

where f_k are the values of the time signal at time $t_k = k\,\Delta t$ with $k = 0, ..., N-1$, N is the number of known time signal values used to do the transform, and F_n are the transformed values.

These transforms reveal periodicities in input data as well as the relative strengths of any periodic component [38], allowing this way to gain insight into the frequency components that are predominant in a given signal. The previous is accomplished by plotting the amplitude of each of the previously stated sinusoidal terms versus its frequency.

To carry out the fast Fourier transform, several facts have to be taken into account to avoid the distortion of the signal. For example (among others):

- To avoid the distortion denominated as *aliasing* (an effect that causes different signals to become indistinguishable when sampled), an appropriate sampling rate f_s (Hz) has to be chosen to not lose information of the original signal due to its discretisation.
- To avoid the distortion denominated as *leakage* (which is manifested as a loss of detail or resolution in the transformed signal), an appropriate sub-sequence length has to be chosen to minimise that effect. The previous sub-sequence of data is usually obtained by filtering the original data with a window function.

7.2.2 Analysis

In the present case, the Fast Fourier transform only has been executed on the tank's water surface height signal at the points in Table 7.1 for the Mesh 5 case, as it is supposedly the most accurate results of the five studied cases (as it is the most refined case). As commented before in Section 7.1, only the last part of said signals have been transformed, starting at the 30.8 s spot with a sample rate of 100 Hz.

The resulting plots are represented in Figures 7.9, 7.10, 7.11, 7.12 and 7.13.

In these figures there are two things to highlight. On first instance, on all the figures there are present two frequency peaks, one at the configured wave's frequency ($f_w = 0.627\,Hz$) and another one at a frequency equal to exactly the double of the configured wave's frequency, $1.25\,Hz$. This is an indication that a complex wave field is created in the simulation's tank due to reflection. So, the waves in the tank have a first and a second harmonic. Reflection is again detected. Secondly, the other information that stands out of these figures is the fact that the amplitude of the first harmonic oscillates. At 2.25 m (Figure 7.9) its amplitude is of almost 0.032 m; at 3.00 m (Figure 7.10), this amplitude decreases; at 3.75 m (Figure 7.11), its value stays more or less the same; but then, at 4.50 m (Figure 7.12) this first harmonic amplitude increases again; and, finally, at 5.25 m (Figure 7.13) decreases.

Normally, if the wave traveled to a horizon situated in infinity, this amplitude would only decrease, because the incoming wave would be losing energy. The same thing if there was no reflection in the wave tank. Consequently, it can be concluded that in the CFD simulation there is wave reflection in the wave tank. Not only that but the fact that the amplitude of the first harmonic frequency oscillates means that in the tank there is probably what is known as a partial standing wave (explained in Section 7.3). To confirm these this hypothesis, the harmonic study of the tank's water surface's height was carried out in Section 7.3.

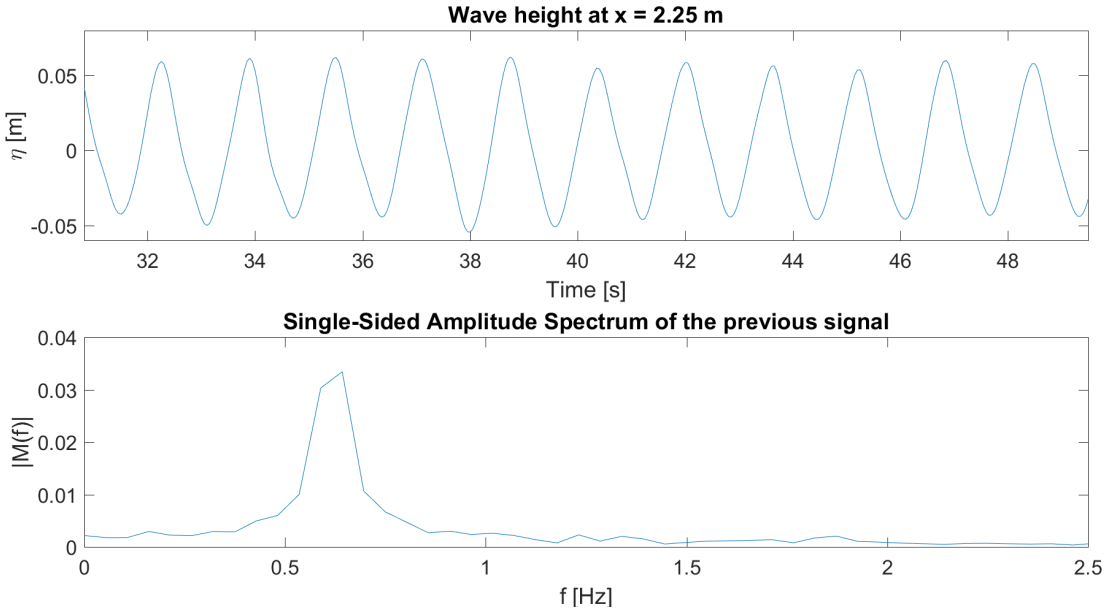

Figure 7.9: Fast Fourier transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 5.

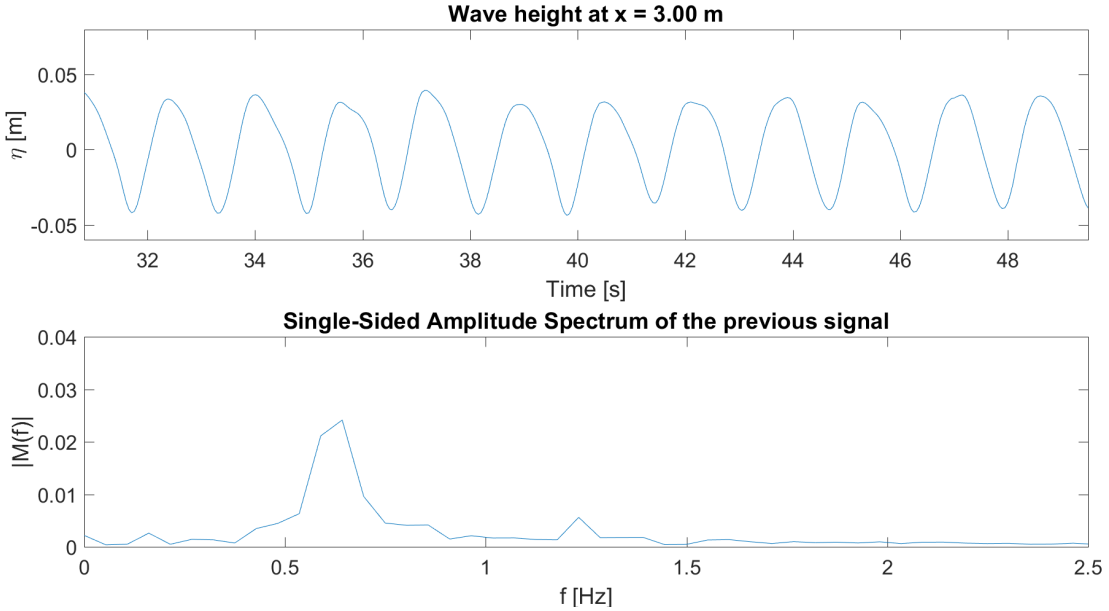

Figure 7.10: Fast Fourier transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 5.

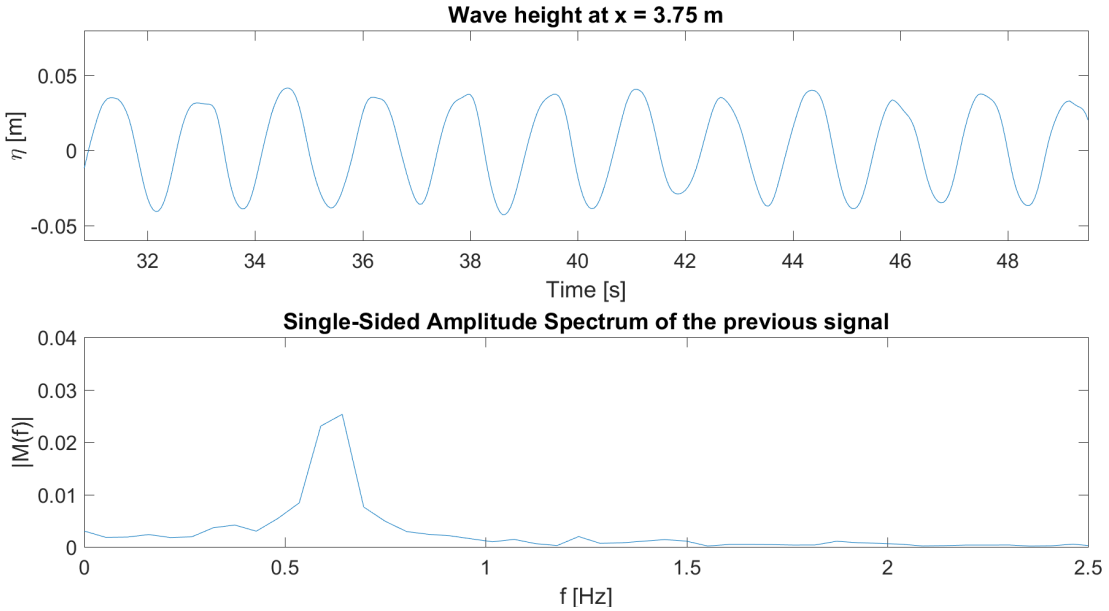

Figure 7.11: Fast Fourier transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 5.

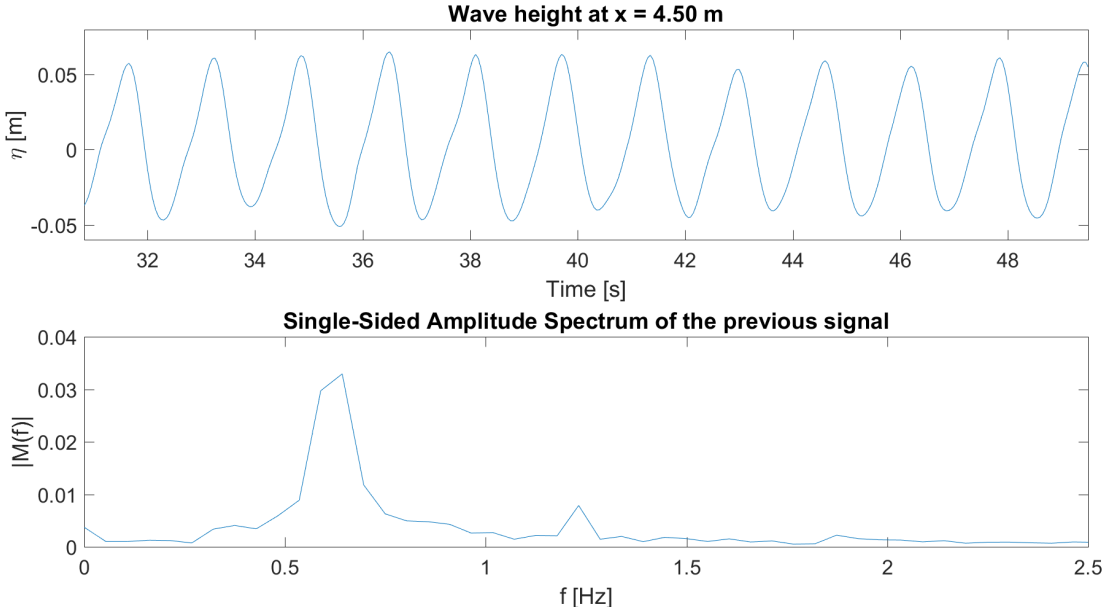

Figure 7.12: Fast Fourier transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 5.

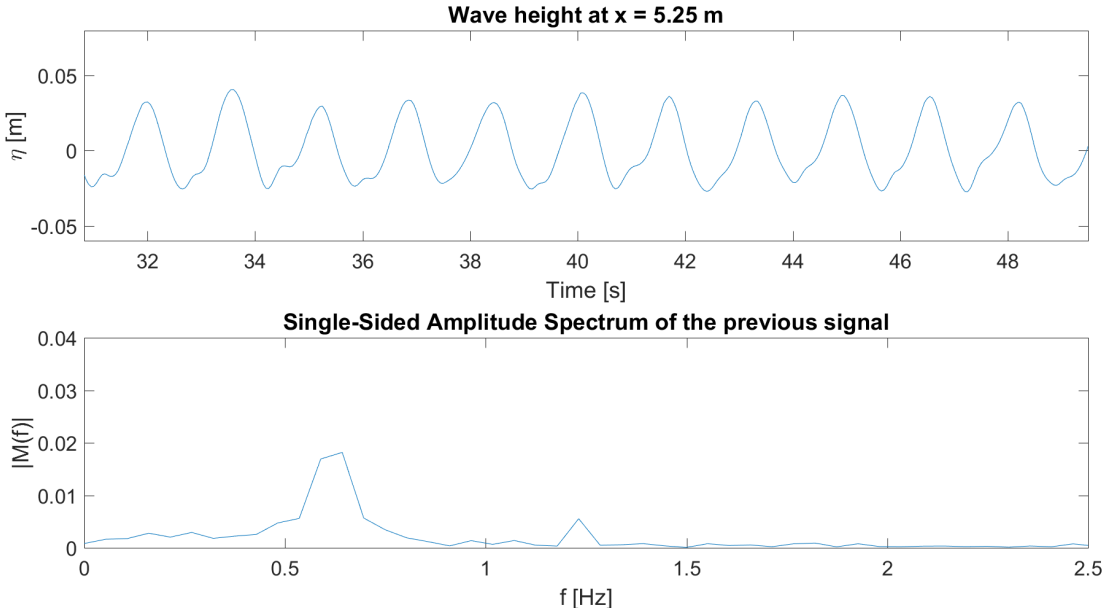

Figure 7.13: Fast Fourier transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 5.

7.3 Harmonic analysis

7.3.1 Introduction

The Harmonic analysis consists on decoupling a specific signal into its monochromatic harmonic components. This way each of the harmonic components can be analysed. The expression used to carry out this decoupling is

$$\eta_i \simeq a_0 + \sum_{j=1}^{N} a_j \, cos(j\sigma t_i) + b_j \, sin(j\sigma t_i) \tag{7.4}$$

which describes the surface elevation in a given location η_i based on a set of coefficients a_j and b_j up to a given order N [39]. Then, $\sqrt{a_j^2 + b_j^2}$ corresponds to the amplitude of the j harmonic component at that specific location. This amplitude is the one represented on the plots of this section.

This decoupling has been carried out for locations inside a range that goes from the 0 m coordinate until the 6.35 m.

7.3.2 Analysis

The harmonic analysis of the surface height at said range of coordinates for Mesh 5 is represented in Figure 7.14.

In this figure, the amplitude of the first and second harmonic of the water surface height is represented. Taking a look at the behaviour of the first harmonic, we can see that its amplitude oscillates in a pseudo-sinusoidal progression along the tank. This fact confirms not only the reflection in the tank, but the presence of a partial standing wave. On the other hand, the second harmonic's amplitude progression is more anarchic and shows the non-linearity of the simulation's wave field. This second harmonic could be originated by multiple factors like reflection or diffusion.

In Figure 7.15, there are also represented the harmonic analysis for Meshes 3 and 4 together with Mesh 5. The first harmonic of the 3 cases doesn't match either, meaning that the solution doesn't converge. Even more than that, it means that the reflection in each of the simulated cases is different, that the amount of reflection generated is mesh dependant, making it impossible for the simulation to converge.

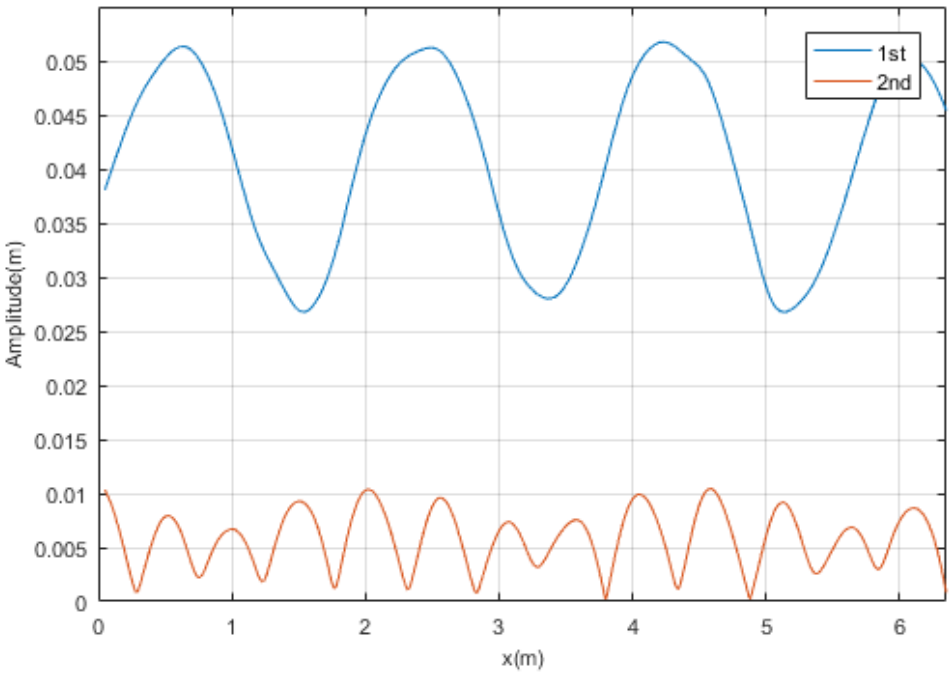

Figure 7.14: Harmonic analysis of the tank's water surface height for $x = 0$ to $x = 6.35\ m$. Mesh 5.

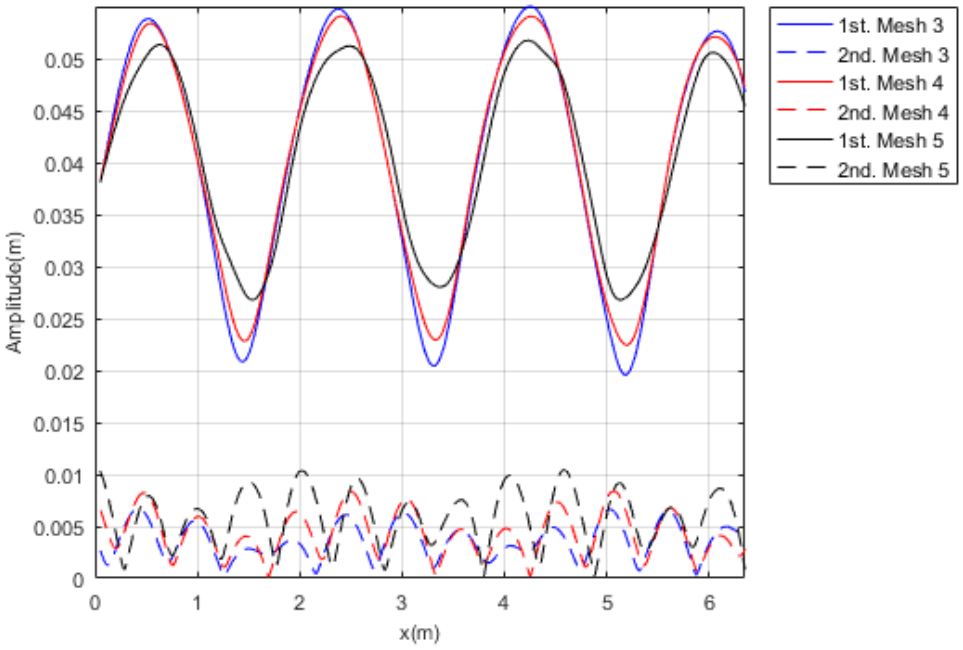

Figure 7.15: Harmonic analysis of the tank's water surface height for $x = 0$ to $x = 6.35\ m$. Meshes 3, 4 and 5.

7.3.3 Partial Standing wave

A standing wave is the combination of two waves moving in opposite directions, each having the same amplitude and frequency [40]. When the amplitude of one of the waves is inferior to the other wave's amplitude is when a partial standing wave is originated.

In this simulations, the incoming wave (the one generated with the wave paddle) moves to the end of the tank. When the wave hits the beach, a part of this incoming wave is reflected back to the tank with the same celerity, but in the opposite direction. The combination of these two waves originate the observed partial standing wave pattern in the first harmonic of the wave height signals.

In Figure 7.16, a partial standing wave and its two components (the incoming and the reflection wave) are shown. More precisely, what can be seen represented is the evolution of the waves' amplitude with space. In the case of the partial standing wave though, the range in which the wave height oscillates at each location is drawn.

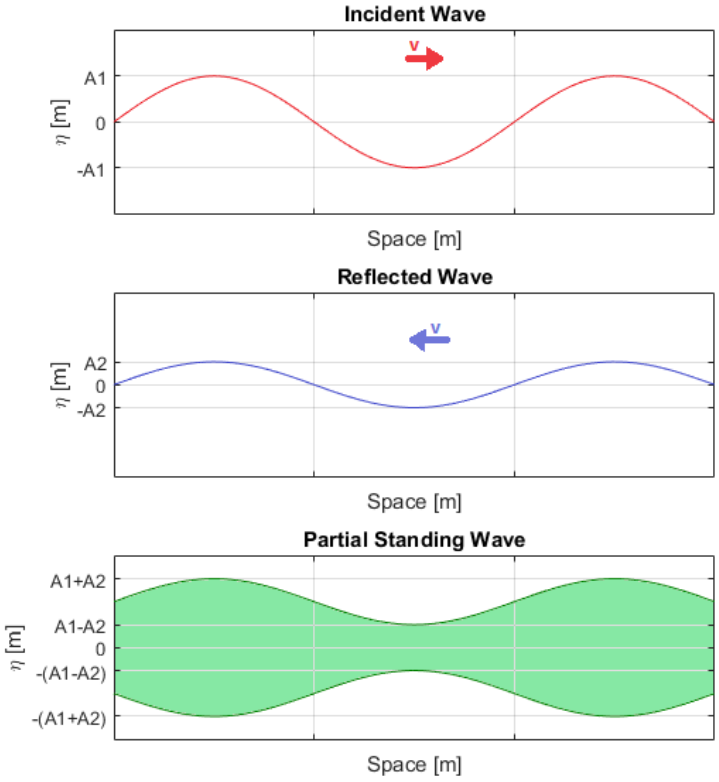

Figure 7.16: Partial Standing Wave

So, in a partial standing wave, the observed amplitude of oscillation of the water surface is different at every coordinate of the tank. The points in the wave tank where the maximum amplitude of the surface height oscillation is reached are known as antinodes, while the points in which the amplitude is minimum are called nodes.

If we take into account the theoretical values of the amplitude in the nodes and antinodes, the data shown in Figure 7.15 can be used to estimate numerically the amount of reflection present in each simulation, or what is the same, the reflection coefficient Γ defined as in (7.5).

$$\Gamma = \frac{A2}{A1} \tag{7.5}$$

Below, the reflection coefficient for Mesh 3, Mesh 4 and Mesh 5 is calculated. The resulting values are different for each Mesh. This confirms the fact that the reflection at each Mesh is different and, consequently, the reflection in the simulations depend on the size of the mesh cells. Thus, making impossible the convergence of the simulation results.

Mesh 3

$$\left.\begin{array}{l} A1 + A2 = 0.0539 \\ A1 - A2 = 0.0203 \end{array}\right\} \quad \left.\begin{array}{l} A1 = 0.0371 \\ A2 = 0.0168 \end{array}\right\} \quad \Gamma_{Mesh3} = \frac{A2}{A1} = 0.4528$$

Mesh 4

$$\left.\begin{array}{l} A1 + A2 = 0.0534 \\ A1 - A2 = 0.0223 \end{array}\right\} \quad \left.\begin{array}{l} A1 = 0.0383 \\ A2 = 0.0160 \end{array}\right\} \quad \Gamma_{Mesh4} = \frac{A2}{A1} = 0.4178$$

Mesh 5

$$\left.\begin{array}{l} A1 + A2 = 0.0512 \\ A1 - A2 = 0.0272 \end{array}\right\} \quad \left.\begin{array}{l} A1 = 0.0392 \\ A2 = 0.0120 \end{array}\right\} \quad \Gamma_{Mesh5} = \frac{A2}{A1} = 0.3061$$

Chapter 8

Conclusions and Further Research

8.1 Conclusions

The main goal of this thesis was to validate the experimental measurements previously taken of the wave energy extraction in the wave tank experimental setup by Jenifer Brenes [27]. To succeed in that effort, a CFD simulation of the same experimental setup was executed with the ComFLOW software. Only one of the experimental scenarios in [27] has been studied, the only one that ComFLOW can simulate with the actual version, the scenario where no floater blanket is present in the wave tank.

This validation attempt was not entirely successful. However, the reasons behind this non-validation have been discovered, and a better insight on the physical phenomena in the wave tank has been achieved.

In the first place, the kinetic energy contained in the seven studied zones for each of the meshes was studied and compared. The comparison between these results led to the conclusion that the simulation results do not converge, even though Mesh 5 is a very refined mesh.

Comparing the average energy content per zone with the experimental measurements' results allowed to determine that, even though the various simulation and experimental results do not match quantitatively, results do match qualitatively showing the same trend in the evolution of the average kinetic content along the seven studied areas in both simulations and reality.

Secondly, a strong reflection has been detected in the wave tank for all the simulations. This was first detected in the time domain analysis of the results and confirmed with the help of the frequency domain analysis tools later used (Wavelet transform plots, Fast Fourier plots and Harmonic analysis).

In the third place, a very specific physical phenomena has been detected in the wave tank, a partial standing wave. This partial standing wave has been demonstrated to have different characteristics at each of the simulations. These differences are caused by the fact that each simulation has a different reflection ratio. Consequently, The amount of reflection in the ComFLOW simulations depends on the size of the mesh.

Finally, it is concluded that the reason behind the non-convergence of the simulation is the mesh-dependency of the wave reflection. As if the amount of reflection in the different mesh simulations was the same, the results would probably converge to a unique solution. Furthermore, the reason

behind the fact that the simulation results do not match quantitatively with the experimental measurements could well be due to a different reflection ratio in both the simulation and the real tanks.

8.2 Further Research

Consequently, further research is needed. The following paths can be pursued, among many others.

On the one hand, a quantification and study of the reflection in the real wave tank experimental setup by means of the water surface's height study along the tank. This way a better insight of the tank's physical phenomena would be accomplished and a better comparison with the simulation results could be done.

On the other hand, the simulation and validation of the *FB* scenario of Jenifer Brenes' thesis. This is when the Floater Blanket is present in the experimental setup. When the ComFLOW simulation software is finally able to simulate complex floating bodies like the Floater Blanket, the next logical step would be to try to validate these results.

In this scenario, the simulation would probably converge, as the floater blanket includes another element that prevents reflection. This new element could be included in the simulation resulting in the reduction of the amount of reflection, thus facilitating the simulation convergence.

Furthermore, as the reflection could be near zero, the experimental results and the simulation results maybe could be finally validated.

Bibliography

[1] C. S. Jones and S. P. Mayfield. *Our Energy Future: Introduction to Renewable Energy and Biofuels*. University of California Press, Oakland, California (USA), 2016.

[2] C. Bellard, C. Bertelsmeier, P. Leadley, W. Thuiller, and F. Courchamp. Impacts of climate change on the future of biodiversity. *Ecology Letters*, January 2012.

[3] British Petroleum. Statistical review of world energy. Technical report, British Petroleum, 2017.

[4] O. Ellaban, H. Abu-Rub, and F. Blaabjerg. Renewable energy resources: Current status, future prospects and their enabling technology. *Renewable and Sustainable Energy Reviews*, November 2014.

[5] U.S. Energy Information Administration. International energy outlook. Technical report, U.S. Energy Information Administration, 2016.

[6] A.K. Akella, R.P. Saini, and M.P. Sharma. Social, economical and environmental impacts of renewable energy systems. *Renewable Energy*, February 2009.

[7] REN21. Renewables global status report. Technical report, REN21, 2017.

[8] Strategic Energy Technologies Information System (SETIS). Ocean energy. Technology Information Sheet, European Commission.

[9] Enerdata. Global energy statistic yearbook. `https://yearbook.enerdata.net/electricity/world-electricity-production-statistics.html`, 2017. [Online; accessed 07-August-2017].

[10] R. Pelc and R. M. Fujita. Renewable energy from the ocean. *Marine Policy*, November 2002.

[11] B. Drew, A. R. Plummer, and M. N. Sahinkaya. A review of wave energy converter technology. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, December 2009.

[12] A. Clément, P. McCullen, A. Falcão, A. Fiorentino, F. Gardner, K. Hammarlund, G. Lemonis, T. Lewis, K. Nielsen, S. Petroncini, M. T. Pontes, B. O. Schild, P. Sjöström, H. C. Søresen, and T. W Thorpe. Wave energy in europe: current status and perspectives. *Renewable and Sustainable Energy Reviews*, October 2002.

[13] T. W. Thorpe. A brief review of wave energy. Technical report, The UK Department of Trade and Industry, 1999.

[14] K. Gunn and C. Stock-Williams. Quantifying the global wave power resource. *Renewable Energy*, August 2012.

[15] M. Alvarez, G. Iglesias, and P. García. Wave energy converters. Technical report, University of Santiago de Compostela, 2001.

[16] Archimedes wave swing. http://www.awsocean.com/technology.html. [Online; accessed 11-August-2017].

[17] Aquamarine power oyster. http://subseaworldnews.com/2012/03/05/uk-aquamarine-power-unveils-proposals-for-lewis-wave-energy-project/. [Online; accessed 11-August-2017].

[18] Wavegen limpet. http://www.esru.strath.ac.uk/EandE/Web_sites/01-02/RE_info/wavecase.htm. [Online; accessed 11-August-2017].

[19] Wave dragon. http://www.wavedragon.net/index.php?option=com_content&task=view&id=6&Itemid=5. [Online; accessed 11-August-2017].

[20] Ocean power technology's powerbuoy. http://www.oceanpowertechnologies.com/. [Online; accessed 11-August-2017].

[21] Pelamis wave power. http://www.pelamiswave.com/pelamis-technology/. [Online; accessed 11-August-2017].

[22] Ocean Grazer Group. The Ocean Grazer: A novel ocean energy collection and storage device. http://www.oceangrazer.com/, 2017. [Online; accessed 13-August-2017].

[23] A.I. Vakis, W. A. Prins, and H. Meijer. First steps in the design and construction of the Ocean Grazer. *Macromolecular Symposia*, July 2014.

[24] H. Meijer. Simulation of a piston-type hydraulic pump for the Ocean Grazer. Master's thesis, University of Groningen, 2014.

[25] M. van Rooij. Experimental validation of dynamical contact models of the Ocean Grazer. Master's thesis, University of Groningen, 2015.

[26] Antonis I Vakis and John S Anagnostopoulos. Mechanical design and modeling of a single-piston pump for the novel power take-off system of a wave energy converter. *Renewable Energy*, 96:531–547, 2016.

[27] J. Brenes Casasola. Energy extraction analysis of the MP^2PTO system and floater blanket of the Ocean Grazer WEC. Bachelor's thesis, University of Groningen, 2017.

[28] L. H. Holthuijsen. *Waves in oceanic and coastal waters*. Cambridge University Press, Cambridge, England (UK), 2007.

[29] The Open University. *Waves, Tides and Shallow-Water Processes*. Butterworth-Heinemann Books, Oxford, England (UK), 1999.

[30] R. Luppes, P. van der Plas, B. Iwanowski, T. Bunnik, B. Duz, H. van der Heiden, R. Wemmenhove, P. Wellens, A. Veldman, T. Helmholt-Kleefsman, E. Loots, and J. Helder. *Manual ComFLOW version 3.9.X/4.0*. University of Groningen, November 2015.

[31] M.J.J. Bögels. Validating floater blanket models for the Ocean Grazer. Bachelor's thesis, University of Groningen, 2017.

[32] W. Thielicke and E. J. Stamhuis. PIVlab – Towards User-friendly, Affordable and Accurate Digital Particle Image Velocimetry in MATLAB. *Journal of Open Research Software*, October 2014.

[33] W. Thielicke and E. J. Stamhuis. PIVlab - Time-Resolved Digital Particle Image Velocimetry Tool for MATLAB. https://pivlab.blogspot.nl/, 2017. [Online; accessed 13-August-2017].

[34] COMSOL Inc. Finite Element Mesh Refinement Definition And Techniques. https://www.comsol.com/multiphysics/mesh-refinement, 2017. [Online; accessed 21-August-2017].

[35] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, Pennsylvania (USA), 1992.

[36] L. Chun-Lin. *A Tutorial of the Wavelet Transform*. National Taiwan University, February 2010.

[37] Y. Wei, T. Abadie, A. Henry, and F. Dias. Wave interaction with an Oscillating Wave Surge Converter. Part II: Slamming. *Ocean Engineering*, February 2016.

[38] E. W. Weisstein. Discrete Fourier Transform. http://mathworld.wolfram.com/DiscreteFourierTransform.html, 2017. [Online; accessed 07-September-2017].

[39] N. G. Jacobsen. *A Full Hydro- and Morphodynamic Description of Breaker Bar Development*. PhD thesis, April 2011.

[40] Editors of Encyclopædia Britannica. Standing wave. https://www.britannica.com/science/standing-wave-physics, 2017. [Online; accessed 07-September-2017].

Appendix A

ComFlow simulation input

In this appendix the files used as input for the CFD simulation carried out with ComFLOW are shown. As commented on the main body of this thesis, the following files correspond to the input files of the case denominated 'Mesh 1'. For cases 'Mesh 2' to 'Mesh 5' the only difference is a different value for parameters *imax* and *kmax* of the *comflow.in* file (see section A.3).

A.1 geometry.in

The geometry of the model is configured in this file. This is possible by defining different basic geometrical entities that together form the whole geometry of the problem.

For each entity some basic information has to be introduced. The first line gives information about the type of entity (for example tetrahedron, wedge, brick,etc.), if the interior is solid (if its value is 0) or empty (otherwise), and (optionally) if the object moves (if its value is 1) or not (otherwise), in that order. The following lines determine the coordinates and/or the radii of the element. For more information see ComFLOW's manual [30].

```
3  0  0
6.35      -0.385   -0.895
9.07      -0.385   -0.895
9.07       0.385   -0.895
6.35       0.385   -0.895
6.77      -0.385   -0.475
9.07      -0.385    0.245
9.07       0.385    0.245
6.77       0.385   -0.475
3  0  0
-0.025    -0.385   -0.895
 0.025    -0.385   -0.895
 0.025     0.385   -0.895
-0.025     0.385   -0.895
-0.025    -0.385   -0.84
 0.025    -0.385   -0.84
 0.025     0.385   -0.84
-0.025     0.385   -0.84
3  0  1
-0.025    -0.385   -0.845
 0.025    -0.385   -0.845
 0.025     0.385   -0.845
-0.025     0.385   -0.845
-0.025    -0.385    0.205
 0.025    -0.385    0.205
 0.025     0.385    0.205
-0.025     0.385    0.205
```

A.2 motionobject.in

In the situation that the simulation to be carried out with ComFLOW includes a moving body, the file named *motionobject.in* is needed to determine and configure the motion of said object.

The first variable of the file describes the type of motion of the moving body (for example constant and accelerated motion, sinusoidal motion, etc.). The rest of variables vary its purpose depending on the type of motion, for more information see ComFLOW's manual [30].

```
1  ------------------------------------------------------------------------------
2  initial motion of object 1 (wave flap)
3  motion (2=sinusoidal)
4  2
5  xrot   yrot   zrot
6  0        0       -0.845
7  A1   A2   A3   A4   A5   A6
8  0.0      0.0      0.0      0.0      0.0698   0.0
9  omega1   omega2   omega3   omega4   omega5   omega6
10 0.0      0.0      0.0      0.0      3.87851  0.0
11 phi1   phi2   phi3   phi4   phi5   phi6
12 0.0      0.0      0.0      0.0      3.1415   0.0
13 ------------------------------------------------------------------------------
```

A.3 comflow.in

The input file *comflow.in* contains variables that describe most of the specifications of the simulation, those that are not related to geometry nor its movement. This includes a wide range of parameters, from time parameters to grid parameters or numerical parameters. For a better understanding see ComFLOW's manual [30].

```
-- Title ------------------------------------------------------------------
wave test
---------------------------------------------------------------------------
slosh    movbdy  twph  nproc research
1        2       0     2     0
---------------------------------------------------------------------------
-- domain definition ------------------------------------------------------
xmin     xmax    ymin     ymax    zmin     zmax
-0.5     9.07    -0.385   0.385   -0.895   0.305
---------------------------------------------------------------------------
-- green water parameters -------------------------------------------------
grnwtr
0
---------------------------------------------------------------------------
high     low     length
0.0      0.0     0.0
---------------------------------------------------------------------------
width    a       b
0.0      0.0     0.0
---------------------------------------------------------------------------
-- definition initial liquid configuration --------------------------------
liqcnf lqxmin  lqxmax  lqymin  lqymax  lqzmin  lqzmax
2      -100    100.    -100.   100.    -10.    0.
---------------------------------------------------------------------------
-- definition of incoming wave --------------------------------------------
wave  wvstart  period  wheight  xcrest  waterd  ramp*  order  curr  beta
0     0        0.      0.       0.      0.      0      0      0.    0.
ramp  val1  val2
0     0.    0.
---------------------------------------------------------------------------
-- definition of in- and outflow boundaries -------------------------------
nrio
5
i/o  plane  xmin   xmax   ymin     ymax    zmin   zmax   spec  start  full
101  1      -0.5   -0.5   -100     100     -100   100    0.0   0.0    0.0
101  1      9.07   9.07   -100     100     -100   100    0.0   0.0    0.0
101  2      -100   100    -0.385   -0.385  -100   100    0.0   0.0    0.0
101  2      -100   100    0.385    0.385   -100   100    0.0   0.0    0.0
101  3      -100   100    -100     100     -0.9   -0.9   0.0   0.0    0.0
---------------------------------------------------------------------------
-- partial slip -----------------------------------------------------------
pscnf psl
0     0.0
---------------------------------------------------------------------------
-- absorbing boundary condition -------------------------------------------
bcl bcr  gabc a0     a1     b1     kh1   kh2   alfa1 alfa2
1   1    2    1.386  0.141  0.453  8.64  5.0   0.0   0.0
---------------------------------------------------------------------------
-- definition of numerical beach in positive x-direction ------------------
numbch  dampto* slope  bstart
0       0       0.0    0.0
---------------------------------------------------------------------------
-- physical parameters ----------------------------------------------------
rho1     rho2  mu1     mu2     sigma  theta  patm   gamma
1.027e3  1.0   1.0e-3  1.7e-5  0.0    90.0   1.0e5  1.4
```

```
56 ------------------------------------------------------------------------------
57 -- viscous effects -----------------------------------------------------------
58 turbles limiter wallmodel diffusion~
59 0       0       0         1
60 ------------------------------------------------------------------------------
61 -- grid parameters ----------------------------------------------------------
62 griddef
63 1
64 imax    jmax    kmax    xc      yc      zc      sx      sy      sz
65 304     1       40      0.0     0.0     0.0     1.0     1.0     1.0
66 ------------------------------------------------------------------------------
67 -- numerical parameters -------------------------------------------------------
68 eps     omega*  itmax   alpha   feab0   feab1   feab2   nrintp  linext
69 1.0E-8  1.0001  10000   1.0     0.0     1.0     0.0     5       1
70 ------------------------------------------------------------------------------
71 -- numerical parameters two-phase flow ----------------------------------------
72 imilu extrap restol imptol upwind imprel irhoav itscr droptol droptolbc
73 5     0      1.0E-8 1.0E-3 2      1      1      0     1e-2    1e-6
74 ------------------------------------------------------------------------------
75 -- time parameters/cfl number -------------------------------------------------
76 dt      tmax    dtmax   cfl     cflmin  cflmax  divl
77 0.01    50.0    0.5     1       0.2     0.5     0
78 ------------------------------------------------------------------------------
79 -- free surface methods -------------------------------------------------------
80 vofmth vofcor divl recon~ advect~
81 2      2      0    1      2
82 ------------------------------------------------------------------------------
83 -- gravitation ----------------------------------------------------------------
84 gravx   gravy   gravz   ginrt   finrt
85 0.0     0.0     -9.81   0       0
86 ------------------------------------------------------------------------------
87 -- motion of coordinate system -----------------------------------------------
88 motionframe
89 0
90 amplx   freqx   amply   freqy   amplz   freqz
91 0.0     0.0     0.0     0.0     0.0     0.0
92 ------------------------------------------------------------------------------
93 omex    omey    omez    x0      y0      z0
94 0.0     0.0     0.0     0.0     0.0     0.0
95 ------------------------------------------------------------------------------
96 -- autosave ------------------------------------------------------------------
97 load    nsave
98 0       0
99 ------------------------------------------------------------------------------
100 -- post-processing: snapshots/screen print/center of mass --------------------
101 npm2d   npm3d   compr   nprnt   ntcom
102 0       1000      0     999999  0
103 ------------------------------------------------------------------------------
104 npmslic nyz     nxz     nxy
105 0       0       0       0
106 planeyz
107 planexz
108 planexy
109 ------------------------------------------------------------------------------
110 -- directory name for snapshots ----------------------------------------------
111 pathname snapshot data:
112 data/
113 ------------------------------------------------------------------------------
114 -- fill boxes, force boxes and flux boxes-------------------------------------
115 nfillb  ntfill
116 0       100000
117 xl      xr      yl      yr      zl      zr
118 ------------------------------------------------------------------------------
119 nfrcb   ntfrc
120 0       100000
```

```
xl      xr      yl      yr      zl      zr  itmoves  method
-------------------------------------------------------------------------------
nfluxb  ntflux
0       0
xl      xr      yl      yr      zl      zr
-------------------------------------------------------------------------------
nrelwh  ntrelwh
0       0
xl      xr      yl      yr      zl      zr
-------------------------------------------------------------------------------
-- stream line/particle path ------------------------------------------------
npartp  npartl  npartc  ntpart
0       0       0       0
xpt     ypt     zpt     tstrt                                   <- points
xl      xr      yl      yr      zl      zr      tstrt           <- lines
xc      yc      zc      radius  orient  tstrt                   <- circles
-------------------------------------------------------------------------------
-- monitor points -----------------------------------------------------------
nmntrp  nmntrl  nmntrc  ntmntr
0       0       0       7500
xpt     ypt     zpt             mvp                                 <- points
xl      xr      yl      yr      zl      zr                      <- lines
xc      yc      zc      radius  orient                          <- circles
-------------------------------------------------------------------------------
-- special boxes ------------------------------------------------------------
nrboxes tstart  sbuvw
7       0.0     1
xmin    xmax    ymin    ymax    zmin    zmax
2.803   2.928   -0.385  0.385  -0.285  -0.085
3.103   3.228   -0.385  0.385  -0.285  -0.085
3.403   3.528   -0.385  0.385  -0.285  -0.085
3.703   3.828   -0.385  0.385  -0.285  -0.085
4.303   4.428   -0.385  0.385  -0.285  -0.085
4.603   4.728   -0.385  0.385  -0.285  -0.085
4.903   5.028   -0.385  0.385  -0.285  -0.085
-------------------------------------------------------------------------------
```

Appendix B

Matlab code

In this appendix, all the code used to calculate the kinetic energy content in the wave tank and analyze the simulation behaviour are shown. Eight files were coded in the form of Matlab functions.

read_velocity_parallel.m, *cal_Ek_parallel.m* and *compare_windows.m* serve the purpose of extracting the velocity of the particles in the seven studied Areas along the tank, calculate its Kinetic Energy content and compare the results of the five different studied cases (different meshes).

read_wh.m, wh_point.m, *wavelet_wt.m*, *fft_wt.m* and *harmonic_analysis_wt.m* were designed to analyze the behaviour of the simulation, discover the physical phenomena existing in the wave tank and compare the reflection present for the five different studied cases (different meshes).

B.1 read_velocity_parallel.m

```
function [coord_tot,vel_tot] = read_velocity_parallel(num_boxes)
    %This function is used to read all Special Box output files stored in
    %one 'datasb' folder. It returns the information stored in those files
    %in the form of two new cell array variables named 'coord_tot' and
    %'vel_tot'.
    %
    %INPUT:
    %   - num_boxes: Corresponds to the number of Special Boxes for which
    %   information is stored in the selected 'datasb' folder.
    %
    %OUTPUT:
    %   - coord_tot [m]: Is a cell array variable. Its length is equal to
    %   the number of Boxes present in the selected folder. Each Box has a
    %   cell of the array assigned (Cell number one corresponds to Special
    %   Box number one, etc.).
    %      In each cell, another cell array of length 3 is stored. In each
    %   one of this new cells, an array is stored that corresponds to the X,
    %   Y and Z coordinates of the mesh cells inside that Special Box, in
    %   that order.
    %      For example: coord_tot{2}{3} would correspond to an array of
    %   the Z-coordinates of the grid inside Special Box number 2 and
    %   coord_tot{4}{1} to the X-coordinates of Special Box number 4.
    %
    %   - vel_tot [m/s]: Is a cell array variable. Its length is equal to
    %   the number of Boxes present in the selected folder. Each Box has a
    %   cell of the array assigned.
    %      In each cell, a new cell array is stored. It has 4 columns and a
    %   number of rows equal to the number of files associated with that
    %   Special Box (each file corresponds to a specific time of the
    %   simulation). In the first column, the associated time of the row is
    %   stored. In columns two to four, the velocity fields u (x-direction),
    %   v (y-direction) and w (z-direction) of the mesh inside that Box are
    %   stored in that order.
    %      Each velocity field is a 3-D matrix of the velocity in the
    %   associated direction in the center of the square mesh cell
    %   determined by the coord_tot variable.
    %      For example: vel_tot{1}{2,1} would yield back time step number
    %   2 of Special Box 1. vel_tot{3}{15,2} would yield back the
    %   x-component of velocity in the center of the mesh cells inside Box
    %   3. vel_tot{4}{3,4}(1,1,1) would yield back the z-component of
    %   velocity in the center of the first cell of the mesh inside Box 4.
    %   In this case, the cell of the mesh corresponds to the one that has
    %   the following coordinates: In the x-direction its left and right
    %   coordinates are coord_tot{4}{1}(1) and coord_tot{4}{1}(2),
    %   respectively; in the y-direction, coord_tot{4}{2}(1) and
    %   coord_tot{4}{2}(2); and, in the z-direction its bottom and top
    %   coordinates are coord_tot{4}{3}(1) and coord_tot{4}{3}(2).

    %Select the 'datasb' folder where the Special Box output is located.
    openfold = uigetdir([]);
    openfold_seperate = strsplit(openfold,'\');

    %If the selected folder is not named 'datasb', stop execution.
    if ~strcmp(openfold_seperate(end), 'datasb')
        display('A \datasb folder must be selected. End of execution.')
        return
    end

    %Load ComFlow.
    sf = CMFSnapshot;

    %Calculate the number of files for each single Special Box, ind_max
```

```
63      %(number of simulated iterations).
64      ubication = dir([openfold, '\*.dat']);
65      ind_max = (length(ubication) - 1)/num_boxes;
66
67      %Initialization of variabes.
68      coord_tot = cell(1,num_boxes);
69      vel_tot = cell(1,num_boxes);
70
71      %Each Box information is processed in parallel.
72      parfor s_box=1:num_boxes
73
74          %Generation of the file path for the first file associated with the
75          %Box s_box (File of t=0).
76          fnm = sprintf('Specialbox%03d_%06d.dat',s_box,0);
77          dirfnm = fullfile(openfold,fnm);
78
79
80          %Load Special Box file. If some error occurs, stop execution.
81          res = sf.loadFile(dirfnm);
82          if res~=0
83              warning('Unable to load Special Box. End of execution.');
84          end
85
86
87          %Coordinates of mesh cells in Special Box s_box.
88          coord = cell(3,1);
89          coord{1} = sf.x{1};
90          coord{2} = sf.y{1};
91          coord{3} = sf.z{1};
92          coord_tot{s_box} = coord; %Store coordinates in the cell of
93                                    %coord_tot associated with s_box.
94
95          %Initialize data
96          vel = cell(ind_max,4);
97
98          %Get velocity components u, v & w at t=0
99          vel{1,1} = sf.var.time;
100         vel{1,2} = sf.var.u{1};
101         vel{1,3} = sf.var.v{1};
102         vel{1,4} = sf.var.w{1};
103
104
105         %Loop to get velocity components u, v & w at other t
106         for ind=1:ind_max-1
107
108             %Generation of the file path for the file ind associated with
109             %the Box s_box.
110             fnm = sprintf('Specialbox%03d_%06d.dat',s_box,ind);
111             dirfnm = fullfile(openfold,fnm);
112
113             %Load Special Box file. If some error occurs, stop execution.
114             res = sf.loadFile(dirfnm);
115             if res~=0
116                 warning('Unable to load Special Box');
117             end
118
119             %Get velocity components u, v & w
120             vel{ind+1,1} = sf.var.time;
121             vel{ind+1,2} = sf.var.u{1};
122             vel{ind+1,3} = sf.var.v{1};
123             vel{ind+1,4} = sf.var.w{1};
124
125         end
126         vel_tot{s_box} = vel; %Store velocity fields in the cell of vel_tot
127                               %associated with s_box.
```

```
128     end
129
130     %Close the Special Box file.
131     sf.unloadFile();
132 end
```

B.2 cal_Ek_parallel.m

```
function [Ek_tot_t,Ek_tot_x,Ek_tot_z] = cal_Ek_parallel(coord_tot,vel_tot)
    %This function is used to calculate the Kinetic Energy in every time
    %step for all the areas of study. It returns the results in the form of
    %three new cell array variables named 'Ek_tot_t', 'Ek_tot_x' and
    %'Ek_tot_z'. Where 'Ek_tot_t' is the absolute velocity for each time
    %step of the simulation and 'Ek_tot_x' and 'Ek_tot_z' are its
    %components. In this version of the code the velocity in the
    %y-direction is not considered as the simulations carried out were in
    %2-D.
    %
    %INPUT:
    %   - coord_tot [m]: Is a cell array variable. Its length is equal to
    %   the number of Special Boxes studied. Each Box has a cell of the
    %   array assigned (Cell number one corresponds to Special Box number
    %   one, etc.).
    %      In each cell, another cell array of length 3 is stored. In each
    %   one of this new cells, an array is stored that corresponds to the X,
    %   Y and Z coordinates of the mesh cells inside that Special Box, in
    %   that order.
    %      For example: coord_tot{2}{3} would correspond to an array of
    %   the Z-coordinates of the grid inside Special Box number 2 and
    %   coord_tot{4}{1} to the X-coordinates of Special Box number 4.
    %   (Obtained from read_velocity_parallel)
    %
    %   - vel_tot [m/s]: Is a cell array variable. Its length is equal to
    %   the number of Special Boxes studied. Each Box has a cell of the
    %   array assigned.
    %      In each cell, a new cell array is stored. It has 4 columns and a
    %   number of rows equal to the number of files associated with that
    %   Special Box (each file corresponds to a specific time of the
    %   simulation). In the first column, the associated time of the row is
    %   stored. In columns two to four, the velocity fields u (x-direction),
    %   v (y-direction) and w (z-direction) of the mesh inside that Box are
    %   stored in that order.
    %      Each velocity field is a 3-D matrix of the velocity in the
    %   associated direction in the center of the square mesh cell
    %   determined by the coord_tot variable.
    %      For example: vel_tot{1}{2,1} would yield back time step number
    %   2 of Special Box 1. vel_tot{3}{15,2} would yield back the
    %   x-component of velocity in the center of the mesh cells inside Box
    %   3. vel_tot{4}{3,4}(1,1,1) would yield back the z-component of
    %   velocity in the center of the first cell of the mesh inside Box 4.
    %   In this case, the cell of the mesh corresponds to the one that has
    %   the following coordinates: In the x-direction its left and right
    %   coordinates are coord_tot{4}{1}(1) and coord_tot{4}{1}(2),
    %   respectively; in the y-direction, coord_tot{4}{2}(1) and
    %   coord_tot{4}{2}(2); and, in the z-direction its bottom and top
    %   coordinates are coord_tot{4}{3}(1) and coord_tot{4}{3}(2).
    %   (Obtained from read_velocity_parallel)
    %
    %OUTPUT:
    %   - Ek_tot_t [J]: Is a cell array variable. Its length is equal to
    %   the number of Areas of the wave tank studied. Each Area has a cell
    %   of the array assigned (Cell number one corresponds to Area 0, cell
    %   number two to Area 1, etc.). Area 0 is located inside the area
    %   contained by Special Box 1, but is smaller. Consequently, to do the
    %   calculations of the Kinetic Energy some interpolation has to be
    %   carried out.
    %      Each of those cells contain a Matrix of 2 columns and a number
    %   of rows equal to the number of time steps in vel_tot. For each row,
    %   in column 1 there is the corresponding time and in column 2 the
    %   value of total kinetic energy contained by that area at that time
```

```
63      %   step.
64      %   For example: Ek_tot_t{1}(1,1) would yield back the time of the
65      %   first time frame for Area 0 (Special Box 1). Ek_tot_t{4}(4,2) would
66      %   yield back the total Kinetic Energy present in Area 3 (Special Box
67      %   4) at the 4th time frame analyzed.
68      %
69      %   - Ek_tot_x [J]: Its structure is equal to Ek_tot_t, the only
70      %   difference is that instead of having the information of TOTAL
71      %   Kinetic Energy in a specific Area, stores the X component of said
72      %   Kinetic Energy.
73      %
74      %   - Ek_tot_z [J]: Its structure is equal to Ek_tot_t, the only
75      %   difference is that instead of having the information of TOTAL
76      %   Kinetic Energy in a specific Area, stores the Z component of said
77      %   Kinetic Energy.
78
79      %Initialize variables.
80      Ek_tot_t = cell(size(vel_tot));
81      Ek_tot_x = cell(size(vel_tot));
82      Ek_tot_z = cell(size(vel_tot));
83
84      %Each Area's Kinetic Energy is calculated in parallel.
85      parfor s_box = 1:length(Ek_tot_t)
86
87          %Determine the exact X-coordinates of the studied Area.
88          if s_box == 1 %Area 0
89              xpositions = [2.811,2.920]
90          elseif s_box == 2 %Area 1
91              xpositions = [3.111,3.220]
92          elseif s_box == 3 %Area 2
93              xpositions = [3.411,3.520]
94          elseif s_box == 4 %Area 3
95              xpositions = [3.711,3.820]
96          elseif s_box == 5 %Area 4
97              xpositions = [4.311,4.420]
98          elseif s_box == 6 %Area 5
99              xpositions = [4.611,4.720]
100         elseif s_box == 7 %Area 6
101             xpositions = [4.911,5.020]
102         end
103
104     %The calculation for each single Area has been coded in the function
105     %cal_Ek
106     [Ek_tot_t{s_box},Ek_tot_x{s_box},Ek_tot_z{s_box}] = ...
107         cal_Ek(coord_tot{s_box},vel_tot{s_box},...
108         xpositions(1),xpositions(2),-0.278,-0.0895);
109     end
110
111 end
112
113 function [Ek_t,Ek_x,Ek_z] = cal_Ek(coord,vtot,xmin,xmax,zmin,zmax)
114     %This function is used to calculate the Kinetic Energy in every time
115     %step for one specific area of study. It returns the results in the
116     %form of three new array variables named 'Ek_t', 'Ek_x' and 'Ek_z'.
117     %Where 'Ek_t' is the absolute velocity for each time step of the
118     %simulation and 'Ek_x' and 'Ek_z' are its components. In this version
119     %of the code the velocity in the y-direction is not considered as the
120     %simulations carried out were in 2-D.
121     %
122     %INPUT:
123     %   - coord: Is a cell array of length 3. Each one of this cells
124     %   correspond to an array that contain the coordinates in the X, Y
125     %   and Z direction of the mesh cells inside the studied Special Box,
126     %   in that order.
127     %       For example: coord{1} and coord{3} would correspond to an array
```

```
128     %   of the X-coordinates and Z-coordinates of the grid inside the
129     %   Special Box, respectively. (Obtained from read_velocity_parallel)
130     %   - vtot: Is a cell array of 4 columns and a number of rows equal to
131     %   the number of files associated with that Special Box (each file
132     %   corresponds to a specific time of the simulation). In the first
133     %   column, the associated time of the row is stored. In columns two to
134     %   four, the velocity fields u (x-direction), v (y-direction) and w
135     %   (z-direction) of the mesh inside that Box are stored in that order.
136     %      Each velocity field is a 3-D matrix of the velocity in the
137     %   associated direction in the center of the square mesh cell
138     %   determined by the coord_tot variable.
139     %      For example: vel{2,1} would yield back time step number
140     %   2 of the Special Box. vel{15,2} would yield back the x-component
141     %   of velocity in the center of the mesh cells inside that Box.
142     %   vel{3,4}(1,1,1) would yield back the z-component of velocity in
143     %   the center of the first cell of the mesh inside the Box. In this
144     %   case, the cell of the mesh corresponds to the one that has the
145     %   following coordinates: In the x-direction its left and right
146     %   coordinates are coord{1}(1) and coord{1}(2), respectively; in the
147     %   y-direction, coord{2}(1) and coord{2}(2); and, in the z-direction
148     %   its bottom and top coordinates are coord{3}(1) and coord{3}(2).
149     %   (Obtained from read_velocity_parallel)
150     %   - xmin: Corresponds to the exact smallest coordinate in the X-axis
151     %   of the Area in which the calculation has to be carried out.
152     %   - xmax: Corresponds to the exact highest coordinate in the X-axis
153     %   of the Area in which the calculation has to be carried out.
154     %   - zmin: Corresponds to the exact smallest coordinate in the Z-axis
155     %   of the Area in which the calculation has to be carried out.
156     %   - zmax: Corresponds to the exact highest coordinate in the Z-axis
157     %   of the Area in which the calculation has to be carried out.
158
159
160     %Data
161     Th = 0.001; %Distance of study in the Y-direction[m]
162     rho = 1027; %Density of the water[kg/m3]
163
164     %Coordinates
165     x = coord{1};
166     z = coord{3};
167
168     ind_max = length(vtot(:,1));
169
170     %Initialize variables.
171     Ek_t = zeros(ind_max,2);
172     Ek_x = zeros(ind_max,2);
173     Ek_z = zeros(ind_max,2);
174
175     %Loop to calculate the Kinetic Energy in each time step of the
176     %simulation.
177     for ind=1:ind_max
178
179         %Time and velocity information for that time step.
180         t = vtot{ind,1};
181         u = vtot{ind,2};
182         w = vtot{ind,4};
183
184         [imax,jmax,kmax] = size(u);
185
186         %Initialize local variables
187         Ex = 0;
188         Ez = 0;
189
190         %Loop in which each cell of the studied Special Box is cycled.
191         for i=1:imax
192             for j=1:jmax
```

```
193                 for k=1:kmax
194
195                     %if the current cell is inside the limits of the Area,
196                     %gather cell size and velocity information
197                     if (x(i)>=xmin) && (x(i+1)<=xmax) && (z(k)>=zmin) &&...
198                             (z(k+1)<=zmax)
199                         x_A = x(i+1)-x(i);
200                         z_A = z(k+1)-z(k);
201                         u_A = u(i,j,k);
202                         w_A = w(i,j,k);
203
204                     %if the Area limit is in the middle of the current cell,
205                     %interpole velocities to the center of the part of the
206                     %mesh cell that is inside the Area limits.
207                     elseif (x(i)<xmin) && (x(i+1)>xmin) && (z(k)>=zmin) && ...
208                             (z(k+1)<=zmax)
209                         %Interpole in X. Left side.
210                         x_A = x(i+1)-xmin;
211                         z_A = z(k+1)-z(k);
212                         u_A = itp_lin(mean([x(i+1),x(i)]),mean([x(i+2),x(i+1)
                                ]),u(i,j,k),u(i+1,j,k),mean([x(i+1),xmin]));
213                         w_A = itp_lin(mean([x(i+1),x(i)]),mean([x(i+2),x(i+1)
                                ]),w(i,j,k),w(i+1,j,k),mean([x(i+1),xmin]));
214                     elseif (z(k)<zmin) && (z(k+1)>zmin) && (x(i)>=xmin) && ...
215                             (x(i+1)<=xmax)
216                         %Interpole in Z. Bottom side.
217                         x_A = x(i+1)-x(i);
218                         z_A = z(k+1)-zmin;
219                         u_A = itp_lin(mean([z(k+1),z(k)]),mean([z(k+2),z(k+1)
                                ]),u(i,j,k),u(i,j,k+1),mean([z(k+1),zmin]));
220                         w_A = itp_lin(mean([z(k+1),z(k)]),mean([z(k+2),z(k+1)
                                ]),w(i,j,k),w(i,j,k+1),mean([z(k+1),zmin]));
221                     elseif (x(i)<xmax) && (x(i+1)>xmax) && (z(k)>=zmin) && ...
222                             (z(k+1)<=zmax)
223                         %Interpole in X. Right side.
224                         x_A = xmax-x(i);
225                         z_A = z(k+1)-z(k);
226                         u_A = itp_lin(mean([x(i-1),x(i)]),mean([x(i),x(i+1)]),
                                u(i-1,j,k),u(i,j,k),mean([x(i),xmax]));
227                         w_A = itp_lin(mean([x(i-1),x(i)]),mean([x(i),x(i+1)]),
                                w(i-1,j,k),w(i,j,k),mean([x(i),xmax]));
228                     elseif (z(k)<zmax) && (z(k+1)>zmax) && (x(i)>=xmin) && ...
229                             (x(i+1)<=xmax)
230                         %Interpole in Z. Top side.
231                         x_A = x(i+1)-x(i);
232                         z_A = zmax-z(k);
233                         u_A = itp_lin(mean([z(k-1),z(k)]),mean([z(k),z(k+1)]),
                                u(i,j,k-1),u(i,j,k),mean([z(k),zmax]));
234                         w_A = itp_lin(mean([z(k-1),z(k)]),mean([z(k),z(k+1)]),
                                w(i,j,k-1),w(i,j,k),mean([z(k),zmax]));
235                     elseif (x(i)<xmin) && (x(i+1)>xmin) && (z(k)<zmin) && ...
236                             (z(k+1)>zmin)
237                         %Interpole in X and Z. Bottom-left corner.
238                         x_A = x(i+1)-xmin;
239                         z_A = z(k+1)-zmin;
240                         u_A = itp_bi(mean([x(i),x(i+1)]),mean([z(k),z(k+1)]),
                                mean([x(i+1),x(i+2)]),mean([z(k+1),z(k+2)]),...
241                                 u(i,j,k),u(i,j,k+1),u(i+1,j,k),u(i+1,j,k+1),mean([
                                    xmin,x(i+1)]),mean([zmin,z(k+1)]));
242                         w_A = itp_bi(mean([x(i),x(i+1)]),mean([z(k),z(k+1)]),
                                mean([x(i+1),x(i+2)]),mean([z(k+1),z(k+2)]),...
243                                 w(i,j,k),w(i,j,k+1),w(i+1,j,k),w(i+1,j,k+1),mean([
                                    xmin,x(i+1)]),mean([zmin,z(k+1)]));
244                     elseif (x(i)<xmax) && (x(i+1)>xmax) && (z(k)<zmin) && ...
245                             (z(k+1)>zmin)
```

```
246                         %Interpole in X and Z. Bottom-right corner.
247                         x_A = xmax-x(i);
248                         z_A = z(k+1)-zmin;
249                         u_A = itp_bi(mean([x(i-1),x(i)]),mean([z(k),z(k+1)]),
                                mean([x(i),x(i+1)]),mean([z(k+1),z(k+2)]),...
250                             u(i-1,j,k),u(i-1,j,k+1),u(i,j,k),u(i,j,k+1),mean([
                                    xmax,x(i)]),mean([zmin,z(k+1)]));
251                         w_A = itp_bi(mean([x(i-1),x(i)]),mean([z(k),z(k+1)]),
                                mean([x(i),x(i+1)]),mean([z(k+1),z(k+2)]),...
252                             w(i-1,j,k),w(i-1,j,k+1),w(i,j,k),w(i,j,k+1),mean([
                                    xmax,x(i)]),mean([zmin,z(k+1)]));
253                     elseif (x(i)<xmax) && (x(i+1)>xmax) && (z(k)<zmax) && ...
254                             (z(k+1)>zmax)
255                         %Interpole in X and Z. Top-right corner.
256                         x_A = xmax-x(i);
257                         z_A = zmax-z(k);
258                         u_A = itp_bi(mean([x(i-1),x(i)]),mean([z(k-1),z(k)]),
                                mean([x(i),x(i+1)]),mean([z(k),z(k+1)]),...
259                             u(i-1,j,k-1),u(i-1,j,k),u(i,j,k-1),u(i,j,k),mean([
                                    xmax,x(i)]),mean([zmax,z(k)]));
260                         w_A = itp_bi(mean([x(i-1),x(i)]),mean([z(k-1),z(k)]),
                                mean([x(i),x(i+1)]),mean([z(k),z(k+1)]),...
261                             w(i-1,j,k-1),w(i-1,j,k),w(i,j,k-1),w(i,j,k),mean([
                                    xmax,x(i)]),mean([zmax,z(k)]));
262                     elseif (x(i)<xmin) && (x(i+1)>xmin) && (z(k)<zmax) && ...
263                             (z(k+1)>zmax)
264                         %Interpole in X and Z. Top-left corner.
265                         x_A = x(i+1)-xmin;
266                         z_A = zmax-z(k);
267                         u_A = itp_bi(mean([x(i),x(i+1)]),mean([z(k-1),z(k)]),
                                mean([x(i+1),x(i+2)]),mean([z(k),z(k+1)]),...
268                             u(i,j,k-1),u(i,j,k),u(i+1,j,k-1),u(i+1,j,k),mean([
                                    xmin,x(i+1)]),mean([zmax,z(k)]));
269                         w_A = itp_bi(mean([x(i),x(i+1)]),mean([z(k-1),z(k)]),
                                mean([x(i+1),x(i+2)]),mean([z(k),z(k+1)]),...
270                             w(i,j,k-1),w(i,j,k),w(i+1,j,k-1),w(i+1,j,k),mean([
                                    xmin,x(i+1)]),mean([zmax,z(k)]));
271                     %Otherwise, don't do anything.
272                     else
273                         continue
274                     end
275
276
277                     mass = rho*x_A*Th*z_A;
278                     Ex = Ex + 0.5*mass*u_A.^2;
279                     Ez = Ez + 0.5*mass*w_A.^2;
280
281                 end
282             end
283         end
284
285         %Store results
286         Ek_t(ind,1) = t;
287         Ek_t(ind,2) = Ex + Ez;
288
289         Ek_x(ind,1) = t;
290         Ek_x(ind,2) = Ex;
291
292         Ek_z(ind,1) = t;
293         Ek_z(ind,2) = Ez;
294     end
295
296     %Print information
297     figure
298     plot(Ek_t(:,1),Ek_t(:,2))
```

```
299     title('Kinetic Energy')
300     xlabel('t [s]')
301     ylabel('E_{k} [J]')
302 end
303 
304 
305 function [v] = itp_bi (x1,y1,x2,y2,v11,v12,v21,v22,x,y)
306     %Bilinear interpolation
307 
308     A = [1,x1,y1,x1*y1;...
309          1,x1,y2,x1*y2;...
310          1,x2,y1,x2*y1;...
311          1,x2,y2,x2*y2];
312 
313     b = inv(A).'*[1;x;y;x*y];
314     v = b(1)*v11+b(2)*v12+b(3)*v21+b(4)*v22;
315 
316 end
317 
318 function [v] = itp_lin (x1,x2,v1,v2,x)
319     %Linear Interpolation
320 
321     v = v1 + (v2-v1)*(x-x1)/(x2-x1);
322 
323 end
```

B.3 compare_windows.m

```
function [] = compare_windows(Ek,E_names)
    %This function plots the Kinetic Energy evolution along time of
    %different simulations on the same plot. The function yields back
    %this plot for each of the studied Areas. Moreover, it prints the
    %maximum and minimum value measured on the DPIV experiments for each
    %window. This makes an easier comparison of results of the different
    %simulated cases and the DPIV measurements.
    %
    %INPUT:
    %   - Ek [J]: Is a cell array of the different Kinetic Energy signals
    %   to be represented on the same plot. Each of the members of this
    %   array corresponds to the output of the function cal_Ek_parallel.
    %
    %   - E_names: Is a cell array of strings. Each string is associated
    %   with a member of the Ek cell array.
    %       For example: E_names{1} would be the name printed on the plot
    %   legend for Ek{1}.

    load('JEN.mat')

    for i=1:7;
       compare_Ek({Ek{1}{i},Ek{2}{i}},E_names,MAX_J(i),MIN_J(i),i)
    end
end

function [] = compare_Ek(E_all,E_name_all,min_J,max_J,i_p)


    colors = ['b','r','g','c','m','y'];

    if length(E_all)~=length(E_name_all)
        display('Both input arrays have to be the same length. End of
            execution.')
        return
    end

    if length(E_all)>length(colors)
        display('Surpased the limit of Ek input. Maximum 6 Ek. End of
            execution.')
        return
    end

    figure(i_p)
    title('Absolute Kinetic Energy')
    xlabel('t [s]')
    ylabel('E_{k} [J]')
    hold on


    for i=1:length(E_all)
        plot(E_all{i}(:,1),E_all{i}(:,2),colors(i));
    end

    plot([0,50],[min_J,min_J],'k--')
    plot([0,50],[max_J,max_J],'k--')

    hold off
    legend([E_name_all,{'Minimum measured','Maximum measured'}])
end
```

B.4 read_wh.m

```
function [coord_wh,wh_t] = read_wh()
    %This function is used to read the Snapshot files of a simulation and
    %gathers all information related with the water heights along the tank.
    %
    %OUTPUT:
    %   - coord_wh [m]: Is an array. Corresponds to the x-axis coordinates
    %   of which we know the evolution of the water height. This points are
    %   located in the center of each one of the mesh cells in the water
    %   surface. The more mesh cells, the longer is coord.
    %
    %   - wh_t [m]: Is a cell array of 2 columns and as many rows as
    %   Snapshot files in the selected folder. Each Snapshot file
    %   correspond to one specific time of the simulation. For each row, on
    %   the first column time is stored while on column number 2 there is
    %   an array with the water height at each position of coord at that
    %   time.
    %       For example: wh_t{1,1} yields back the row's time. wh_t{5,2}
    %   yields back an array with the water height along the tank at row 5's
    %   associated time. wh_t{34,2}(3) yields back the water height at
    %   coord(3) at row 34's associated time.


    %Select the 'data' folder in which the Snapshot output is located.
    openfold = uigetdir([]);
    openfold_seperate = strsplit(openfold,'\');

    %If the selected folder is not named 'data', stop execution.
    if ~strcmp(openfold_seperate(end), 'data')
        display('A \data folder must be selected. End of execution.')
        return
    end

    %Load ComFlow
    sf = CMFSnapshot;

    %Calculate the number of Snapshot files, ind_max (number of simulated
    %iterations).
    ubication = dir([openfold, '\*.dat']);
    ind_max = ((length(ubication) - 2));

    %Generation of the file path for the first Snapshot file(File of t=0).
    fnm = sprintf('cmf%1dd%04d.dat',3,0);
    dirfnm = fullfile(openfold,fnm);


    %Load Snapshot file. If some error occurs, stop execution.
    res = sf.loadFile(dirfnm);
    if res~=0
        warning('Unable to load snapshot. End of execution.');
        return
    end

    wh = sf.waterHeight(); %Auxiliar variable

    %Coordinates of the center of the surface mesh cells. The point from
    %where water height information is obtained.
    coord_wh = wh.xc;

    %Initialize data
    wh_t = cell(ind_max,2);

    %Get water heights at t=0
```

```
63      wh_t{1,1} = sf.var.time;
64      wh_t{1,2} = wh.wh;
65
66
67      %Loop to get the water heights at all other t.
68      for ind=1:ind_max-1
69
70          %Generation of the file path for file ind.
71          fnm = sprintf('cmf%1dd%04d.dat',3,ind);
72          dirfnm = fullfile(openfold,fnm);
73
74          %Load Snapshot file. If some error occurs, stop execution.
75          res = sf.loadFile(dirfnm);
76          if res~=0
77              warning('Unable to load snapshot. End of execution.');
78              return
79          end
80
81          wh = sf.waterHeight(); %Auxiliar variable
82
83          %Get water heights
84          wh_t{ind+1,1} = sf.var.time;
85          wh_t{ind+1,2} = wh.wh;
86
87      end
88
89      %Close the snapshot file
90      sf.unloadFile();
91  end
```

B.5 wh_point.m

```
function wh_p = wh_point(x,coord_wh,wh_t)
    %This function interpoles the evolution of the water height at a
    %specific point of the wave tank.
    %
    %INPUT:
    %   - x [m]: The x-axis coordinate of the desired evolution of the
    %   water height.
    %
    %   - coord_wh [m]: Is an array. Corresponds to the x-axis coordinates
    %   of which we know the evolution of the water height. (Obtained from
    %   read_wh)
    %
    %   - wh_t [m]: Is a cell array of 2 columns and as many rows as
    %   Snapshot files stored during ComFLOW's simulation. Each Snapshot
    %   file correspond to one specific time of the simulation. For each
    %   row, on the first column time is stored while on column number 2
    %   there is an array with the water height at each position of coord
    %   at that time. (Obtained from read_wh)
    %
    %OUTPUT:
    %   - wh_p [m]: Is a cell array of 2 columns and as many rows as
    %   Snapshot files in the selected folder. wh corresponds to the water
    %   height evolution at one point of the wave tank. For each row, on
    %   the first column time is stored while on column number 2 the water
    %   height associated to that time is stored.

    i_t_max = size(wh_t(:,1),1);
    wh_p = zeros(i_t_max,2);

    for i_t = 1:i_t_max
        wh_p(i_t,2) = interp1(coord_wh,wh_t{i_t,2},x);
        wh_p(i_t,1) = wh_t{i_t,1};
    end
end
```

B.6 wavelet_wt.m

```
function [] = wavelet_wt(wh_p,T)
    %This function yields back the wavelet diagram of the input water height
    %signal.
    %
    %INPUT:
    %   - wh_p [m]: Is an array of 2 columns and as many rows as Snapshot
    %    files stored during ComFLOW's simulation. Each Snapshot file
    %    correspond to one specific time of the simulation. wh corresponds
    %    to the water height evolution at one point of the wave tank. For
    %    each row, on the first column time is stored while on column number
    %    2 the wave height associated to that time is stored. (Obtained from
    %    wh_point)
    %
    %   - T [s]: Is the theorical period of the wh signal.


    %First, we ensure that we have data on the evolution of water height
    %every 0,01 s.
    t_i = 0:0.01:49.5;
    wh_i = interp1(wh_p(:,1),wh_p(:,2),t_i);
    len_wh_i = length(wh_i);


    Fs = 100; %Frequency of the signal (1/(0,01 s)).
    fc = centfrq('cmor1-1'); %Center frequency of the wavelet function 'cmor1
        -1'.
    freqrange = [0.5/T 5/T]; %Selected frequency range to be plotted.
    scalerange = fc./(freqrange*(1/Fs)); %Scale range to be plotted (
        Transformed from freqrange).

    %Compute coefficients COEFS using cwt
    scales = linspace(scalerange(1),scalerange(end),100);
    Coeffs = cwt(wh_i(1:end-1),scales,'cmor1-1');
    pfreq = scal2frq(scales,'cmor1-1',1/Fs);
    periods = pfreq;

    uT = 6.1646; %Time that a wave spends in traveling from one point of the
        wave tank to both ends and back.

    figure('units','normalized','outerposition',[0 0 1 1]);
    ~ = wscalogram('image',Coeffs,'scales',periods,'xdata',t_i); %Plot diagram
        .

    %Customization of plot
    set(gca,'XTick',0:uT*Fs:len_wh_i);
    set(gca,'XTickLabel',0:1:t_i(end)/uT);
    xlabh = get(gca,'XLabel');
    set(xlabh,'Position',get(xlabh,'Position') - [0 1 0])
    ylabel('Frequency [Hz]')
    xlabel('$\frac{t \cdot c}{2 \cdot 1}$','Interpreter','latex','FontSize'
        ,20)
end
```

B.7 fft_wt.m

```
function [] = fft_wt(wh_p)
    %This function yields back the fourier diagram of the input water height
    %signal.
    %
    %INPUT:
    %   - wh_p [m]: Is an array of 2 columns and as many rows as Snapshot
    %    files stored during ComFLOW's simulation. Each Snapshot file
    %    correspond to one specific time of the simulation. wh corresponds
    %    to the water height evolution at one point of the wave tank. For
    %    each row, on the first column time is stored while on column number
    %    2 the wave height associated to that time is stored. (Obtained from
    %    wh_point)


    uT = 6.1646; %Time that a wave spends in traveling from one point of the
        wave tank to both ends and back.

    %First, we ensure that we have data on the evolution of water height
    %every 0,01 s. Moreover, we discard the first seconds of signal because
    %of its unsteady nature.
    t_i = uT*5:0.01:49.5;
    wh_i = interp1(wh_p(:,1),wh_p(:,2),t_i);
    len_wh_i = length(wh_i);

    Fs = 100; %Frequency of the signal (1/(0,01 s)).
    transform = fft(wh_i);
    P2 = abs(transform/len_wh_i);
    P1 = P2(1:len_wh_i/2+1);
    P1(2:end-1) = 2*P1(2:end-1); %Single-sided amplitude spectrum.
    f = Fs*(0:(len_wh_i/2))/len_wh_i; %Frequency domain.

    %Figure plotting.
    figure
    subplot(2,1,1)
    plot(t_i,wh_i)
    xlim([t_i(1),t_i(end)])
    ylim([-0.06,0.08])
    title('Water height')
    xlabel('Time [s]')
    ylabel('\eta [m]')
    subplot(2,1,2)
    plot(f,P1)
    xlim([0,2.5])
    ylim([0,0.04])
    title('Single-Sided Amplitude Spectrum of the previous signal')
    xlabel('f [Hz]')
    ylabel('|M(f)|')
end
```

B.8 harmonic_analysis_wt.m

```
function [x_N,amp_N] = harmonic_analysis_wt(coord_wh,wh_t)
    %This function yields back the harmonic analysis diagram of the input
    %water height signal.
    %
    %INPUT:
    %   - coord_wh [m]: Is an array. Corresponds to the x-axis coordinates
    %   of which we know the evolution of the water height. This points are
    %   located in the center of each one of the mesh cells in the water
    %   surface. The more mesh cells, the longer is coord. (Obtained from
    %   read_wh)
    %
    %   - wh_t [m]: Is a cell array of 2 columns and as many rows as
    %   Snapshot files stored during ComFLOW's simulation. Each Snapshot
    %   file correspond to one specific time of the simulation. For each
    %   row, on the first column time is stored while on column number 2
    %   there is an array with the water height at each position of coord
    %   at that time. (Obtained from read_wh)
    %
    %OUTPUT:
    %   - x_N [m]: Is an array. Corresponds to the x-axis coordinates along
    %   the tank in which the harmonic analysis of the water height has
    %   been carried out. This points are located in the center of each one
    %   of the mesh cells in the water surface.
    %
    %   - amp_N [m]: Is an array of 2 rows and as many columns as points
    %   analyzed (same length as x_N). Each column corresponds to one of
    %   these coordinates and each row corresponds to the 1st and 2nd order
    %   water height harmonic signals into which wh_t has been decomposed,
    %   respectively.


T=1.62; %Theorical period of the wh signal.
w=2*pi/T; %Associated wave frequency.
n=2; %Maximum harmonic order into which the water height signal will be
    decomposed.

%First, the input information is put in only one matrix array. And the
%coordinates to study are selected (the ones between the wave paddle, at
%0.05 m, and the beach, 6.35 m).
[time_N,data_N] =NumericalWaveData(coord_wh,wh_t);
x_N=coord_wh(find(coord_wh>0.05,1):find(coord_wh>6.35,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Harmonic Analysis Procedure
m=length(time_N);
for ii=1:length(x_N);
    for i=1:m;
        for j=1:n;
            A_N(i,2*j-1)=cos(j*w*time_N(i));
            A_N(i,2*j)=sin(j*w*time_N(i));
        end
    end
    coef_N(:,ii)=A_N\data_N(:,ii);
    data_a_N=A_N*coef_N(:,ii);
    data_a_N_all(:,ii)=data_a_N;
end

%Plots to validate the harmonic analysis aproximation of the original
%signal.
    figure(1);
    plot(time_N,data_N(:,100),'r-');hold on;
```

```
    plot(time_N,data_a_N_all(:,100),'b-');hold on;

    figure(2);
    plot(time_N,data_N(:,200),'r-');hold on;
    plot(time_N,data_a_N_all(:,200),'b-');hold on;


%Calculation of the magnitude of each harmonic signal at each position.
for ii=1:length(x_N);
    for j=1:n
        amp_N(j,ii)=sqrt(coef_N(2*j-1,ii)*coef_N(2*j-1,ii)+...
                         coef_N(2*j,ii)*coef_N(2*j,ii));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plotting of the results
figure(8);
plot(x_N,amp_N(1,:),'-',x_N,amp_N(2,:),'-');hold on;

grid on;
axis([0,6.35,0,0.055]);
xlabel('x(m)');
ylabel('Amplitude(m)');
h=legend('1st','2nd','location','NorthEast');

set(gcf,'PaperUnits','centimeters');
set(gcf,'PaperSize',[15 10]);
set(gcf,'PaperPosition',[0 0 15 10]);
end


function [time,data]=NumericalWaveData(coord_wh,wh)
    %This function outputs the input information in another matlab
    %structure.
    %
    %INPUT:
    %   - coord_wh: Is an array. Corresponds to the x-axis coordinates of
    %   which we know the evolution of the water height. This points are
    %   located in the center of each one of the mesh cells in the water
    %   surface. The more mesh cells, the longer is coord. (Obtained from
    %   read_wh)
    %   - wh_t: Is a cell array of 2 columns and as many rows as Snapshot
    %   files stored during ComFLOW's simulation. Each Snapshot file
    %   correspond to one specific time of the simulation. For each row, on
    %   the first column time is stored while on column number 2 there is
    %   an array with the water height at each position of coord at that
    %   time. (Obtained from read_wh)
    %
    %OUTPUT:
    %   - time: Is an array. Corresponds to the time instants in which wave
    %   height is known.
    %   - data: Is a matrix array that has as many rows as the time
    %   instants of time and as many columns as points in the wave tank in
    %   which the study is carried out. Each cell has stored the wave
    %   heigth at the corresponding time and coordinate.
    %      For example: data(3,5) yields back the water height at time 3 and
    %   position 5.

    i_xmin = find(coord_wh>0.05,1);
    i_xmax = find(coord_wh>6.35,1);

    uT = 6.1646;

    time_aux = cell2mat(wh(:,1));
```

```
127     i_tmin = find(time_aux>5*uT,1);
128     time = time_aux(i_tmin:end);
129
130     data = zeros(length(time),length(coord_wh(i_xmin:i_xmax)));
131
132     for i_t=1:length(time)
133         data(i_t,:)=wh{i_t+i_tmin-1,2}(i_xmin:i_xmax);
134     end
135 end
```

Appendix C

Kinetic energy content plots

In this appendix there are all the Figures that serve the purpose of comparing the results of the different simulated meshes. All meshes have been compared with one another in each area of study. The compared values correspond to the evolution of kinetic energy content in each of the areas.

Area 0

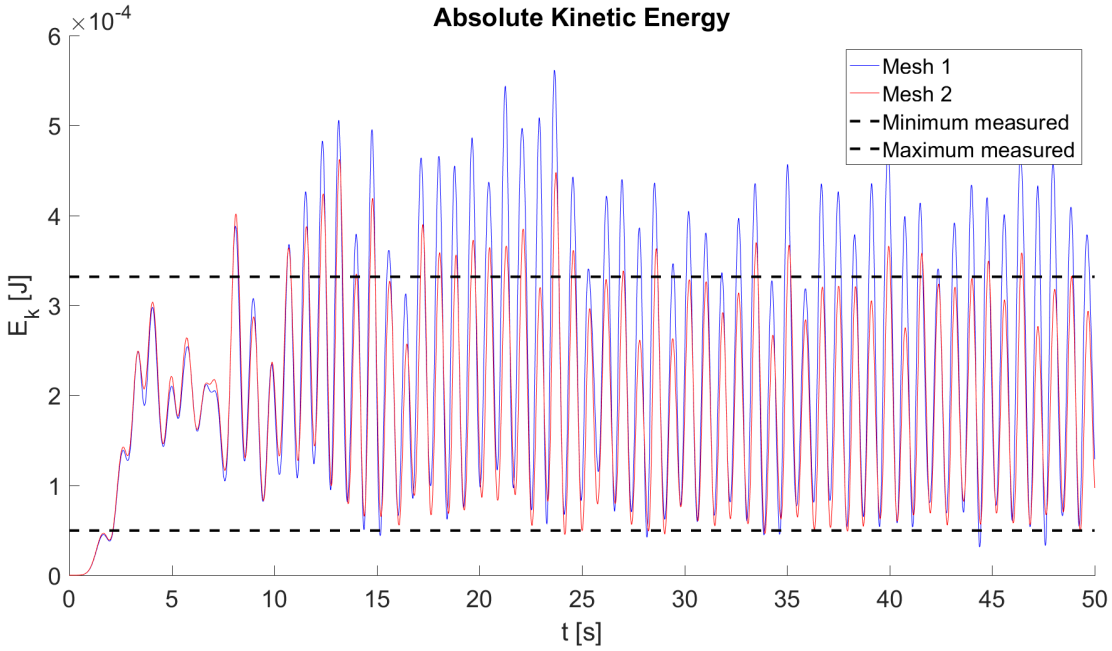

Figure C.1: Kinetic energy content evolution in Area 0 of Mesh 1 and Mesh 2.

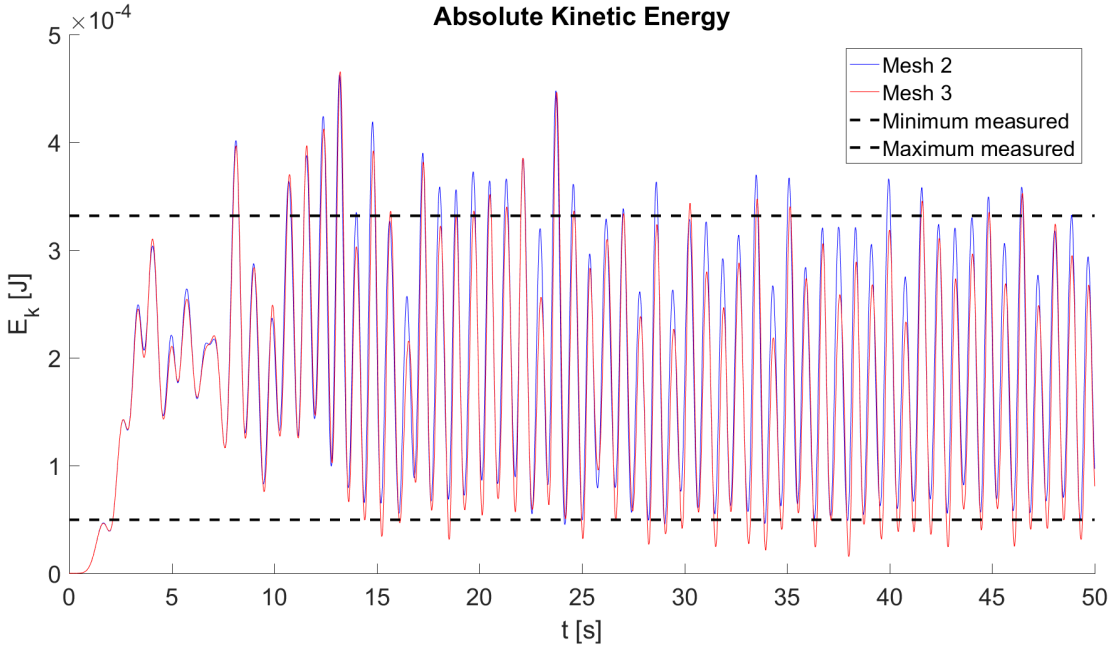

Figure C.2: Kinetic energy content evolution in Area 0 of Mesh 2 and Mesh 3.

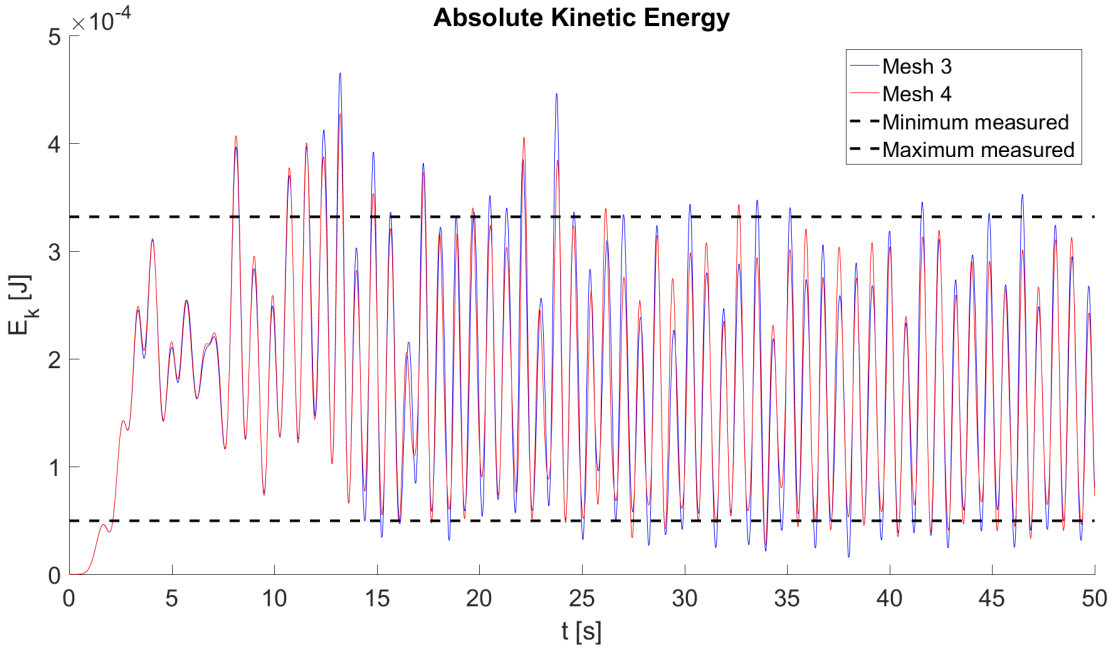

Figure C.3: Kinetic energy content evolution in Area 0 of Mesh 3 and Mesh 4.

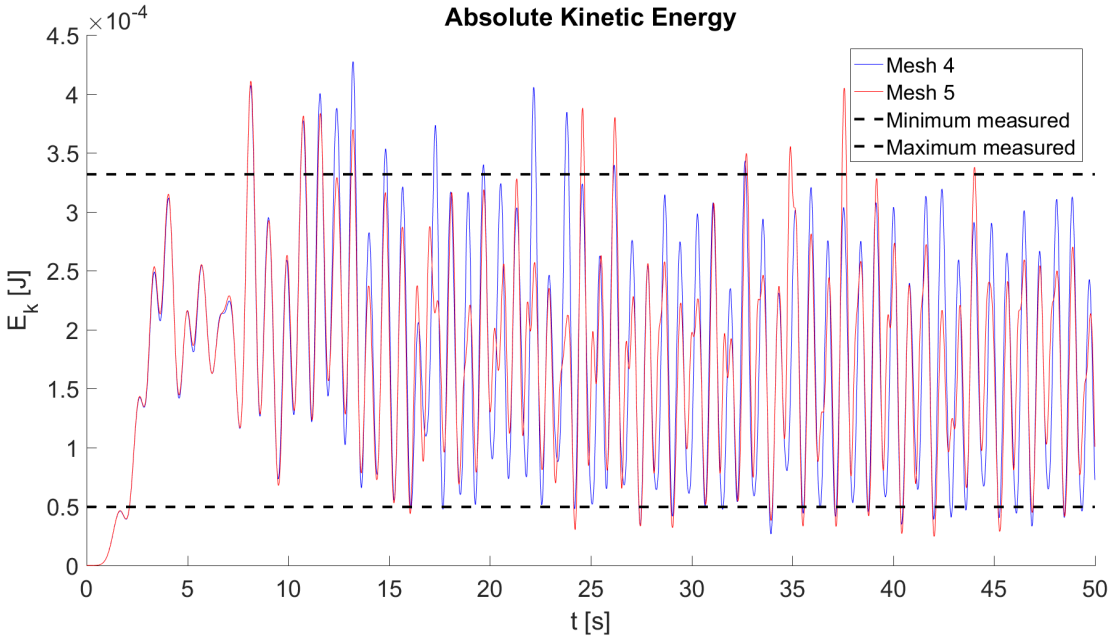

Figure C.4: Kinetic energy content evolution in Area 0 of Mesh 4 and Mesh 5.

Area 1

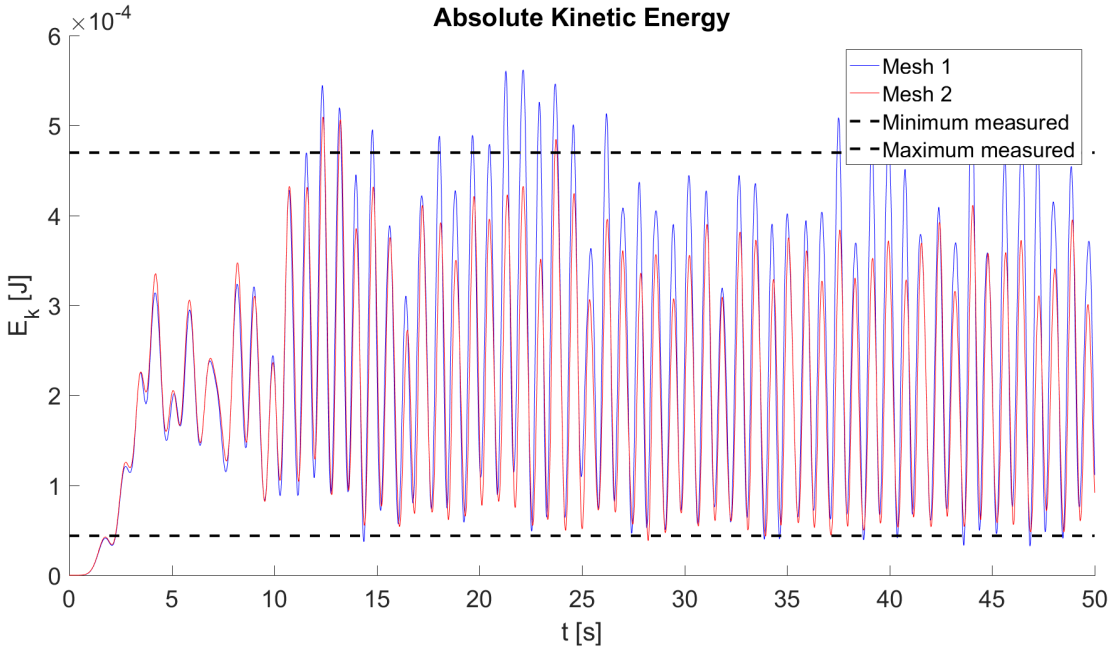

Figure C.5: Kinetic energy content evolution in Area 1 of Mesh 1 and Mesh 2.

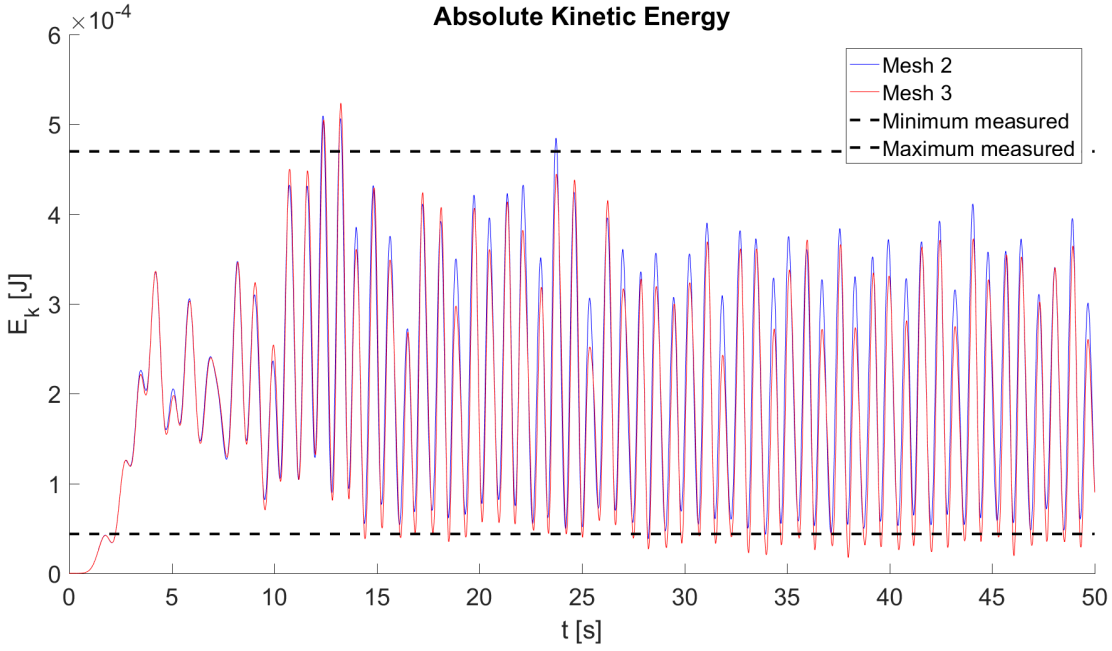

Figure C.6: Kinetic energy content evolution in Area 1 of Mesh 2 and Mesh 3.

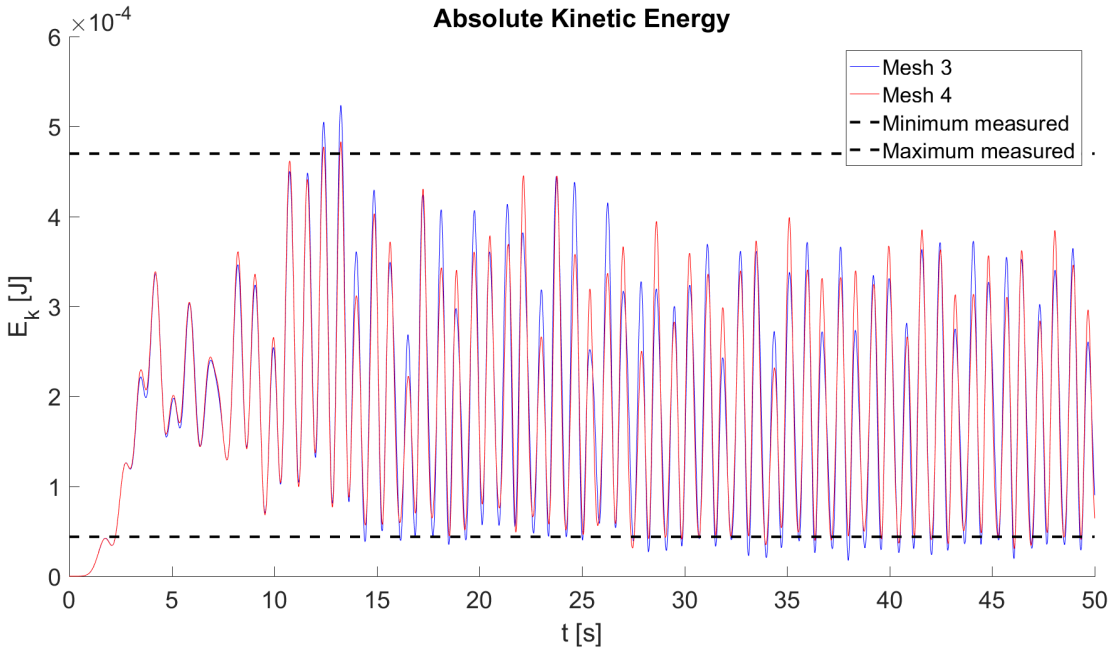

Figure C.7: Kinetic energy content evolution in Area 1 of Mesh 3 and Mesh 4.

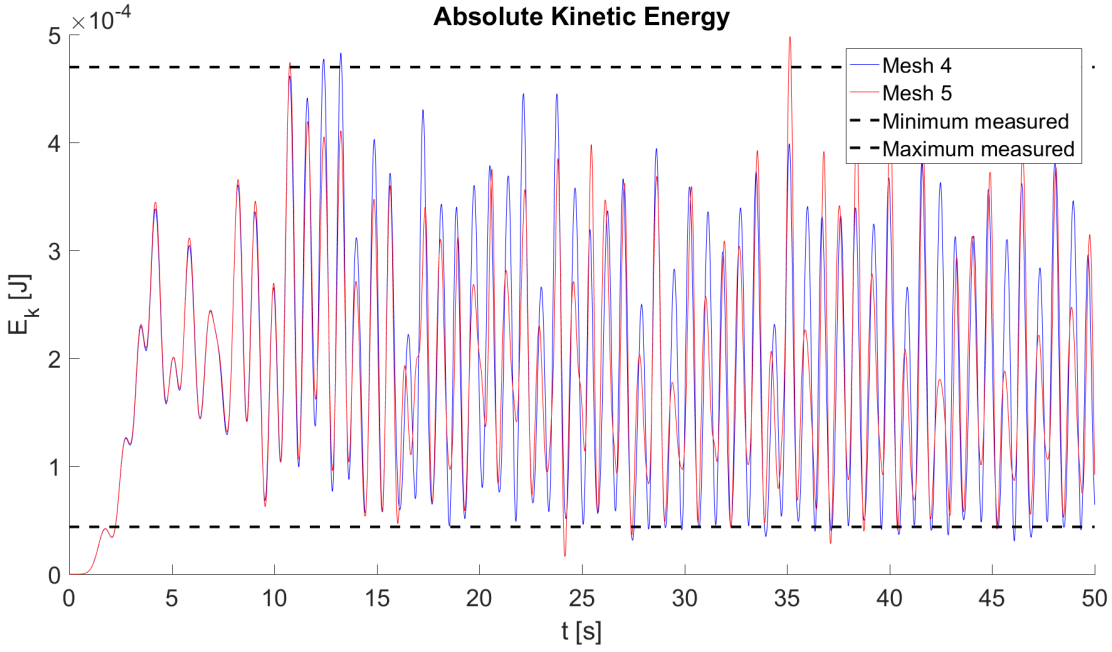

Figure C.8: Kinetic energy content evolution in Area 1 of Mesh 4 and Mesh 5.

Area 2

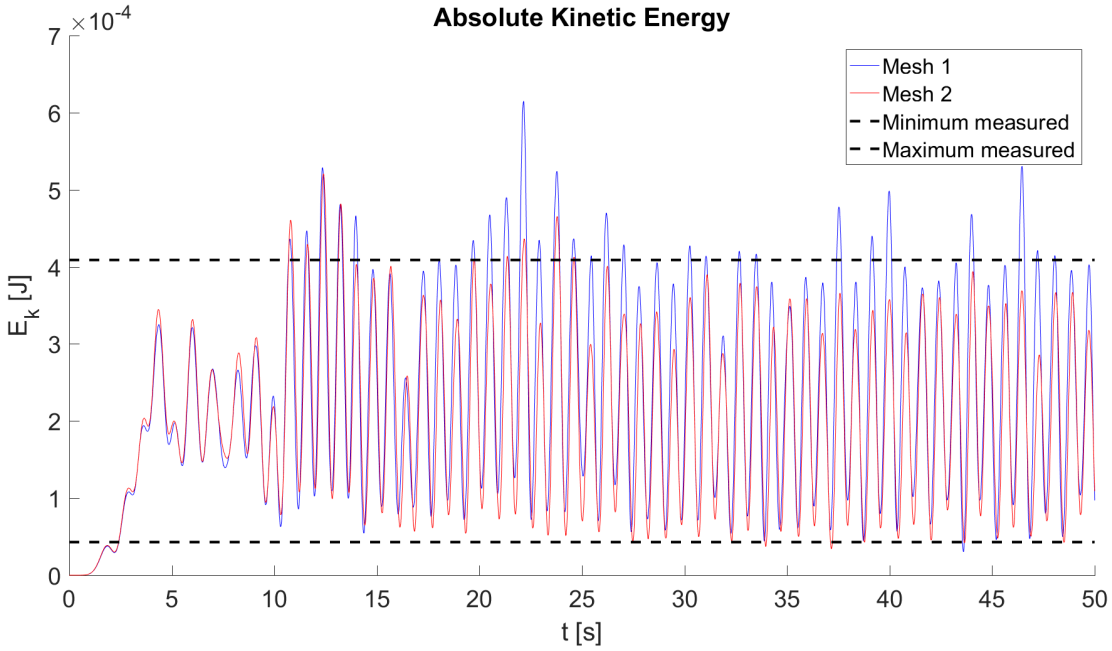

Figure C.9: Kinetic energy content evolution in Area 2 of Mesh 1 and Mesh 2.

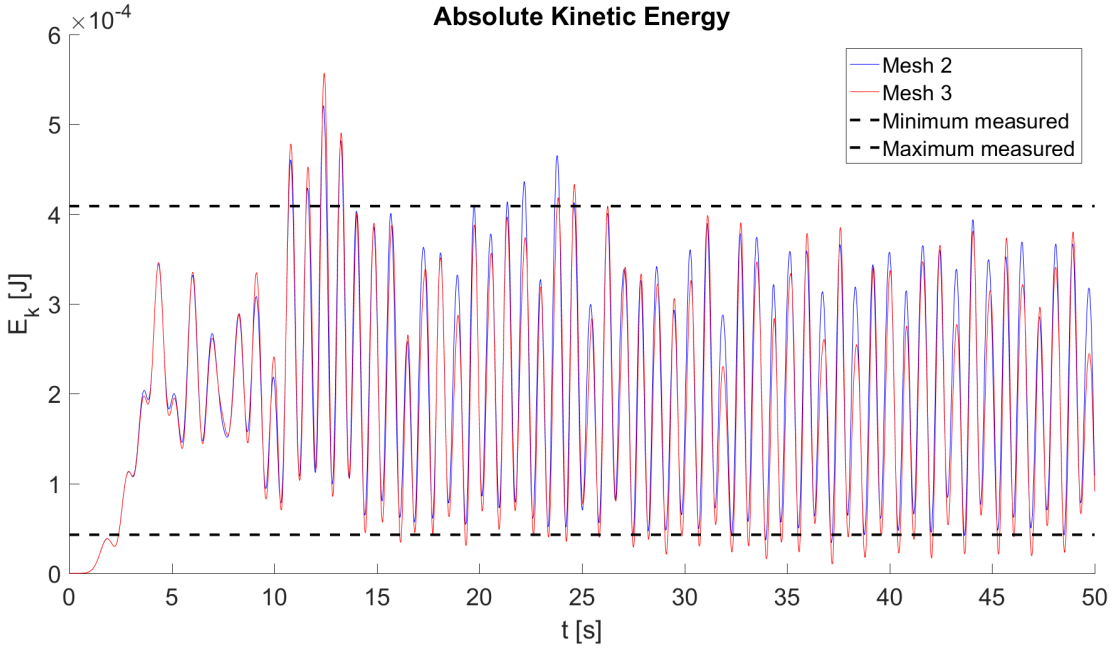

Figure C.10: Kinetic energy content evolution in Area 2 of Mesh 2 and Mesh 3.

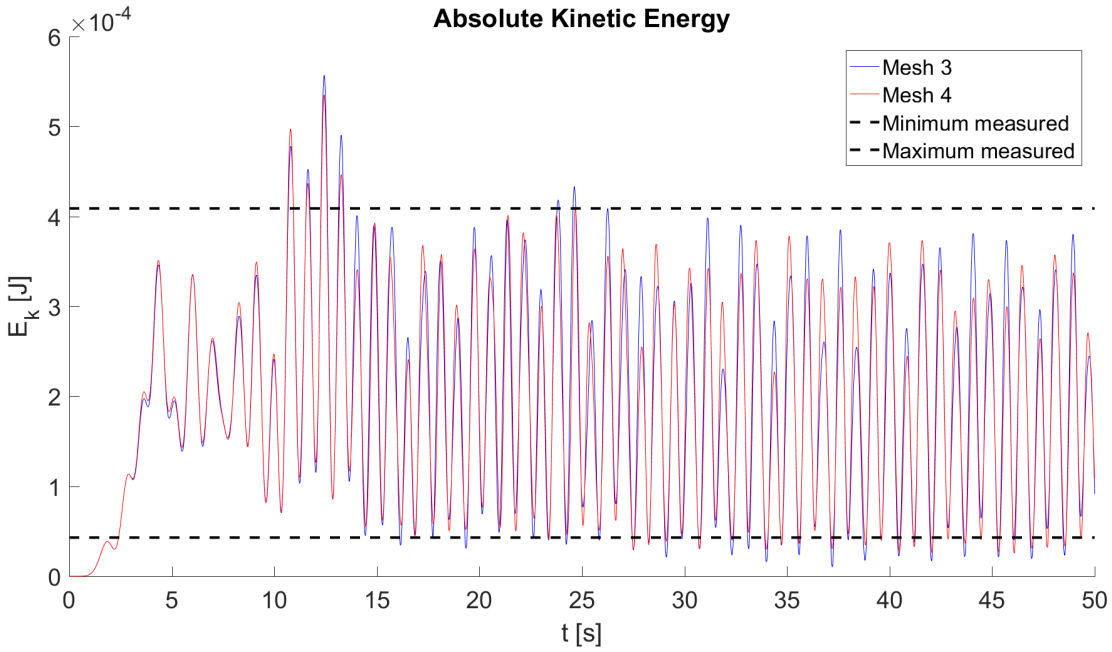

Figure C.11: Kinetic energy content evolution in Area 2 of Mesh 3 and Mesh 4.

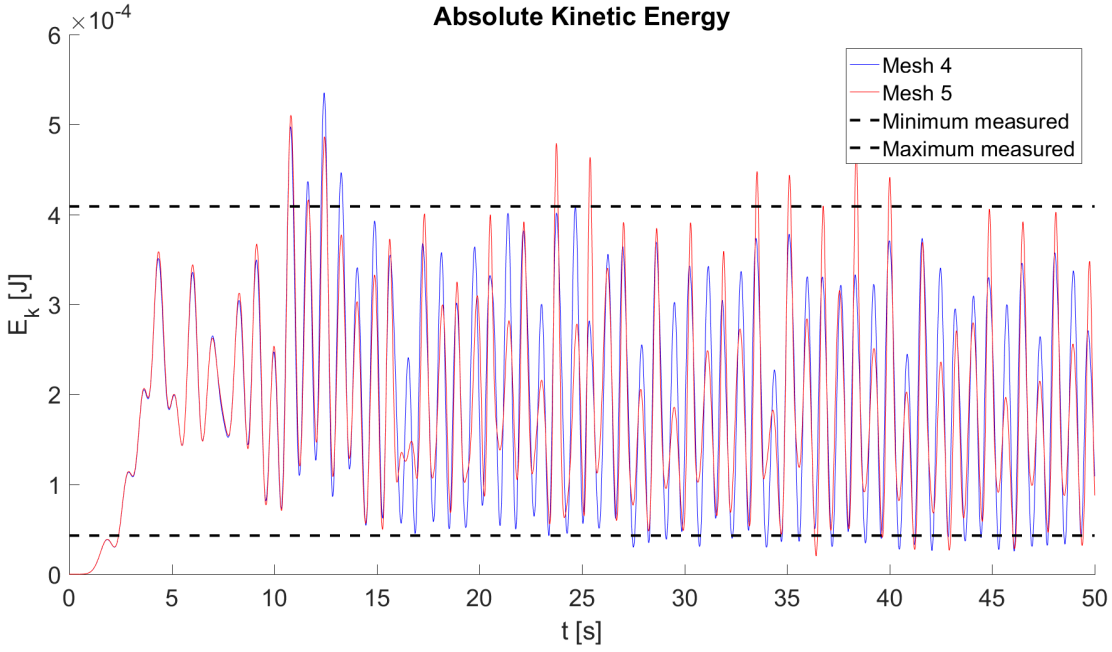

Figure C.12: Kinetic energy content evolution in Area 2 of Mesh 4 and Mesh 5.

Area 3

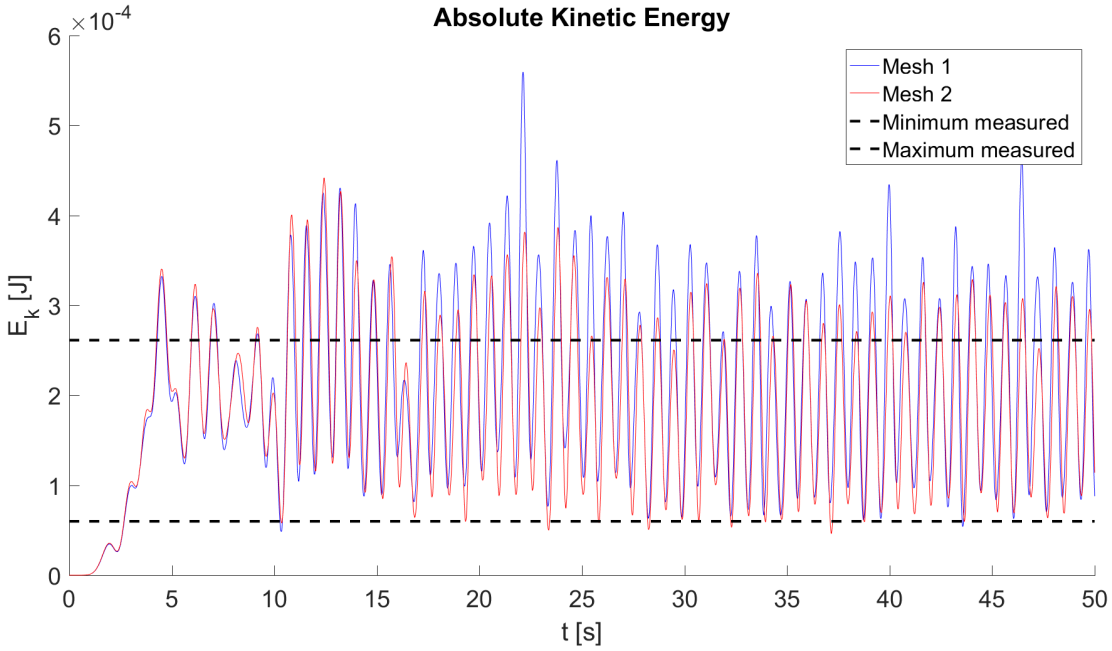

Figure C.13: Kinetic energy content evolution in Area 3 of Mesh 1 and Mesh 2.

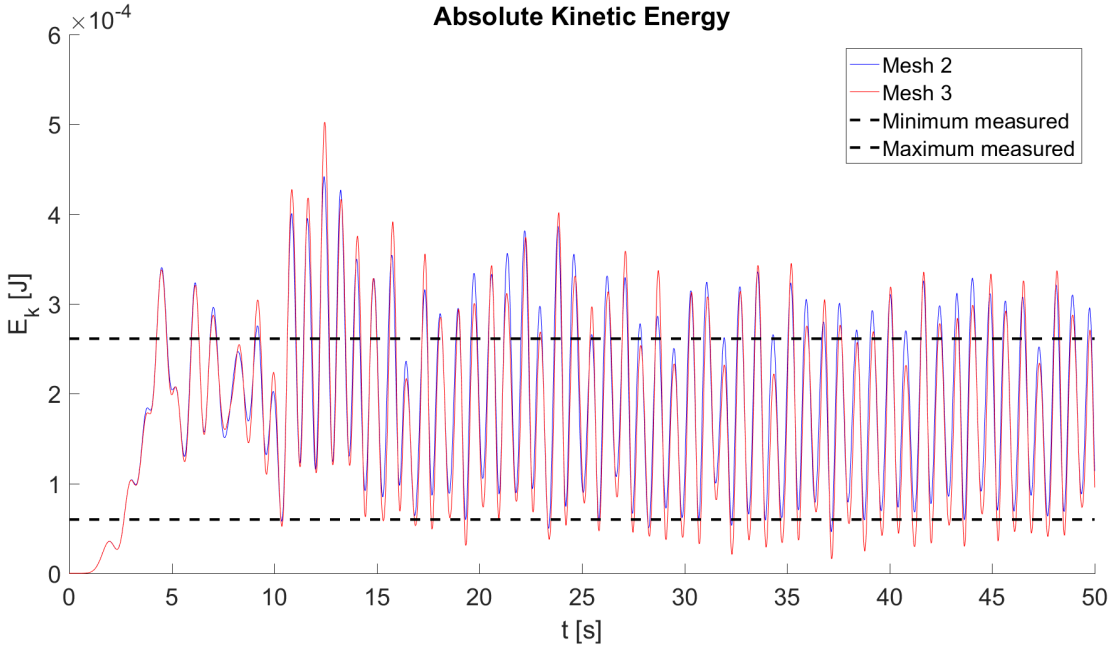

Figure C.14: Kinetic energy content evolution in Area 3 of Mesh 2 and Mesh 3.

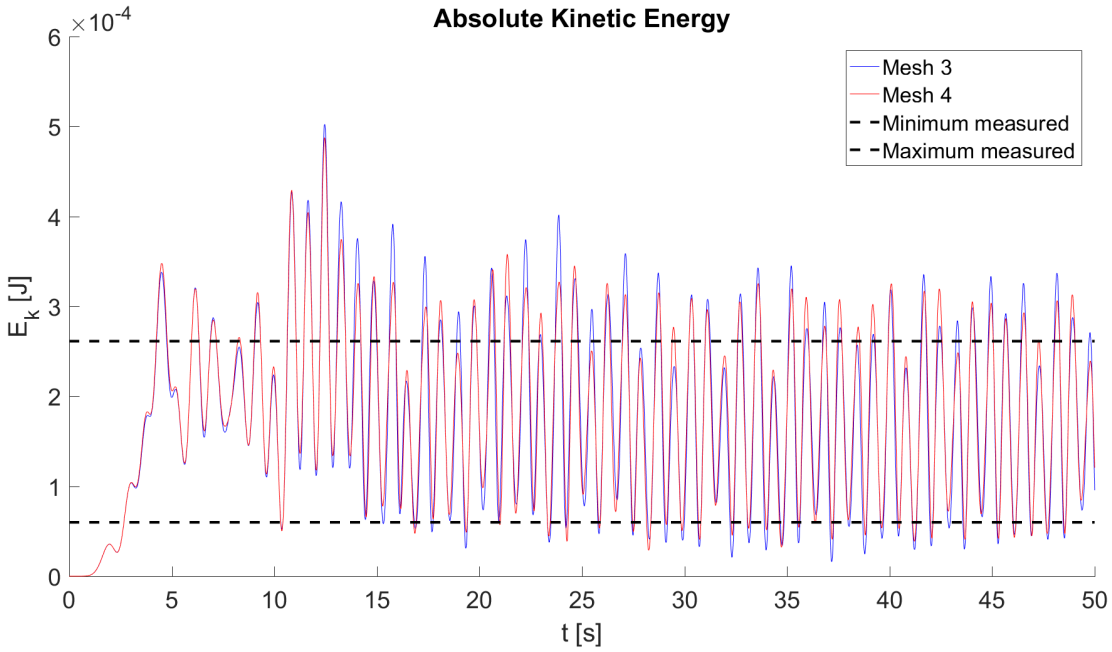

Figure C.15: Kinetic energy content evolution in Area 3 of Mesh 3 and Mesh 4.

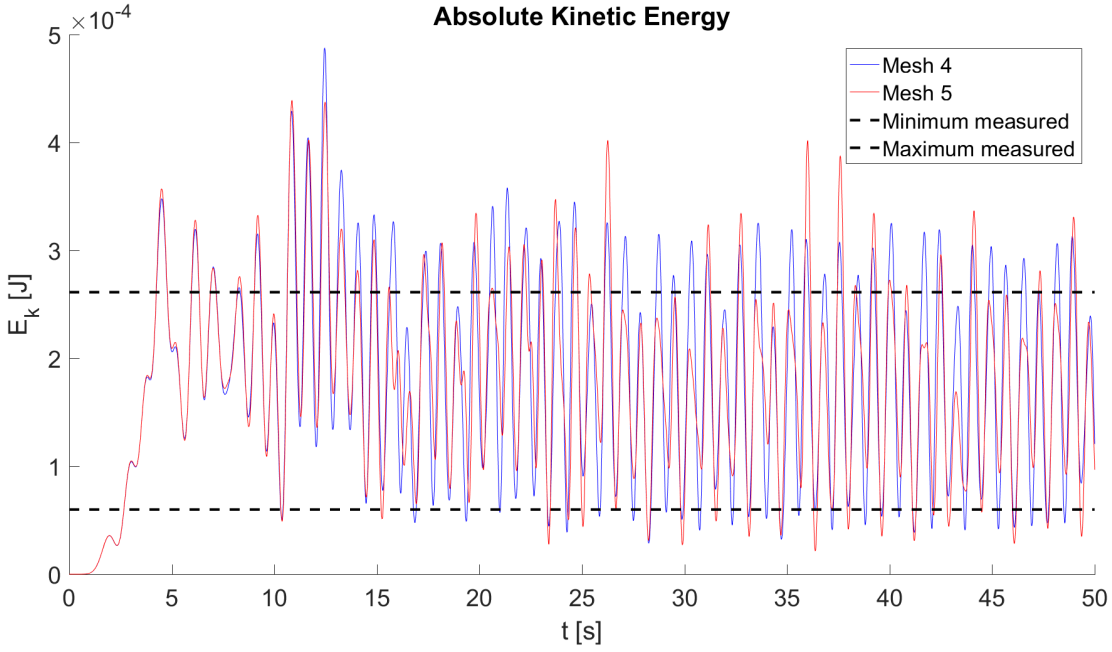

Figure C.16: Kinetic energy content evolution in Area 3 of Mesh 4 and Mesh 5.

Area 4

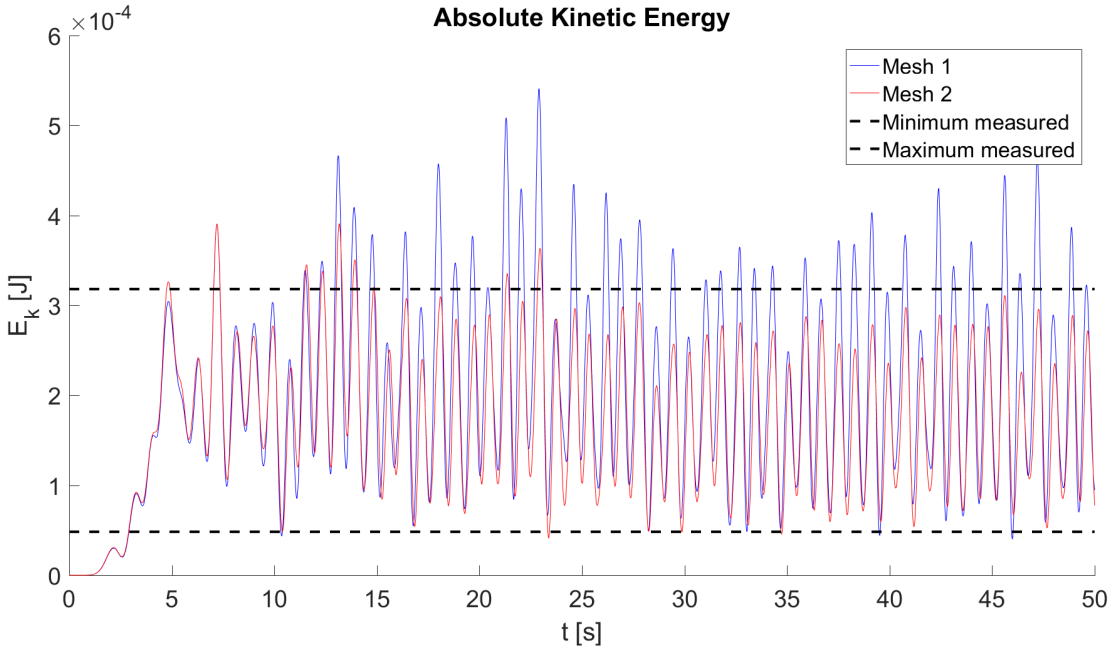

Figure C.17: Kinetic energy content evolution in Area 4 of Mesh 1 and Mesh 2.

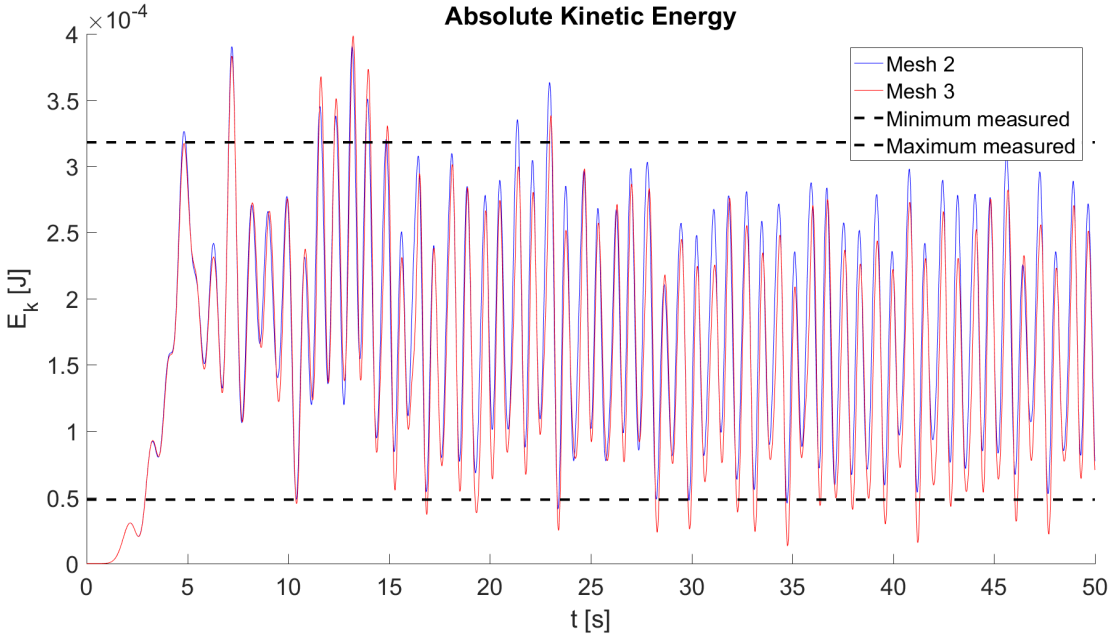

Figure C.18: Kinetic energy content evolution in Area 4 of Mesh 2 and Mesh 3.

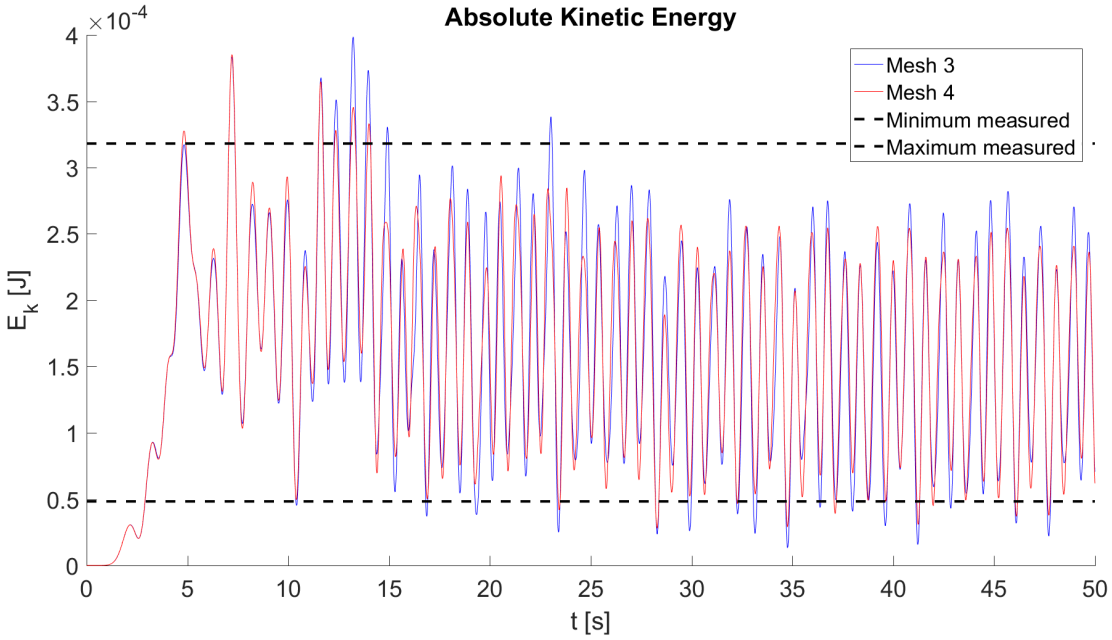

Figure C.19: Kinetic energy content evolution in Area 4 of Mesh 3 and Mesh 4.

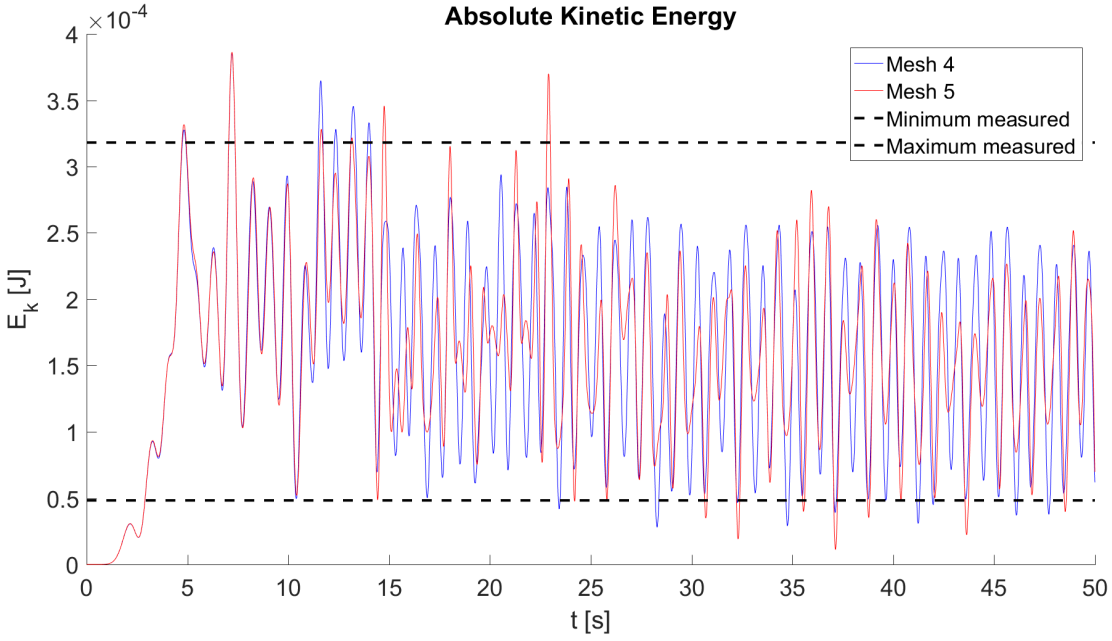

Figure C.20: Kinetic energy content evolution in Area 4 of Mesh 4 and Mesh 5.

Area 5

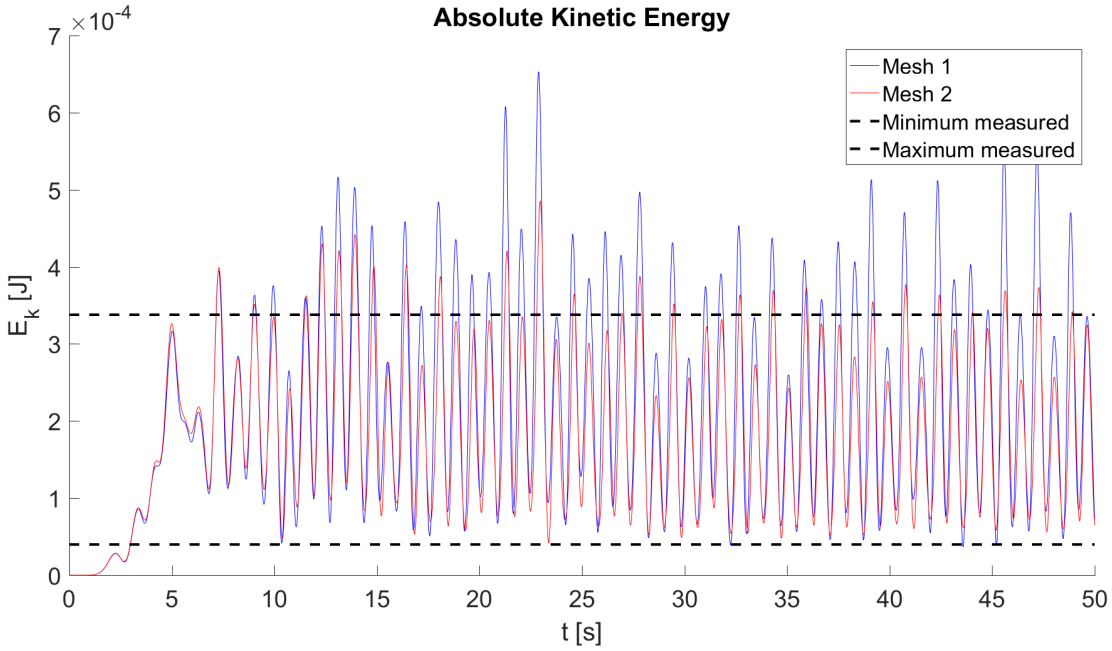

Figure C.21: Kinetic energy content evolution in Area 5 of Mesh 1 and Mesh 2.

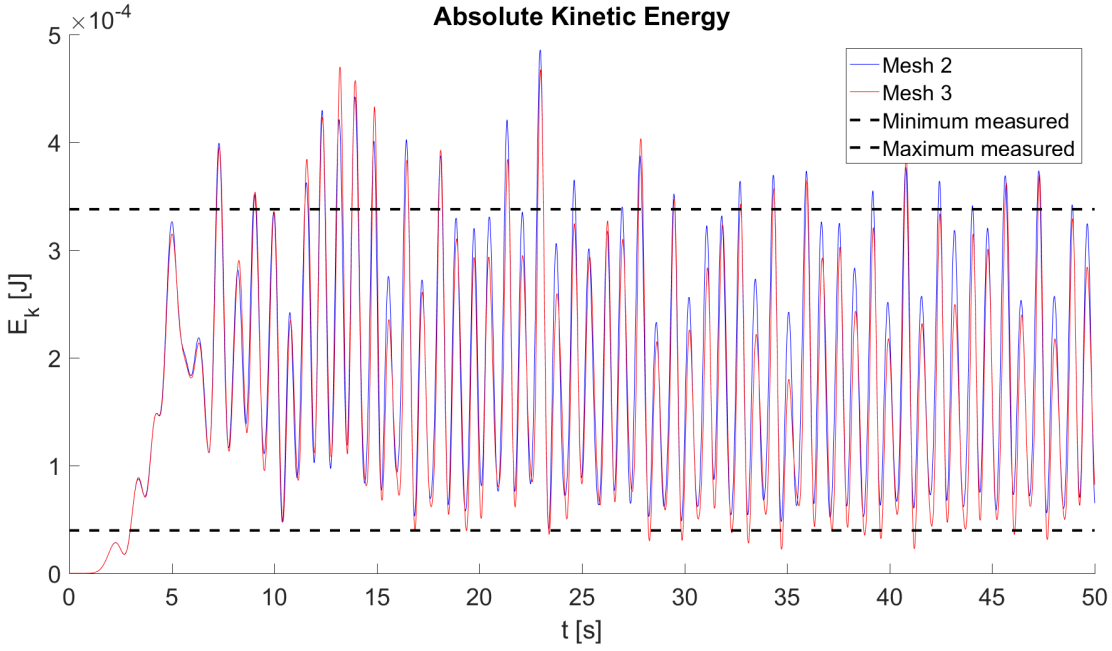

Figure C.22: Kinetic energy content evolution in Area 5 of Mesh 2 and Mesh 3.

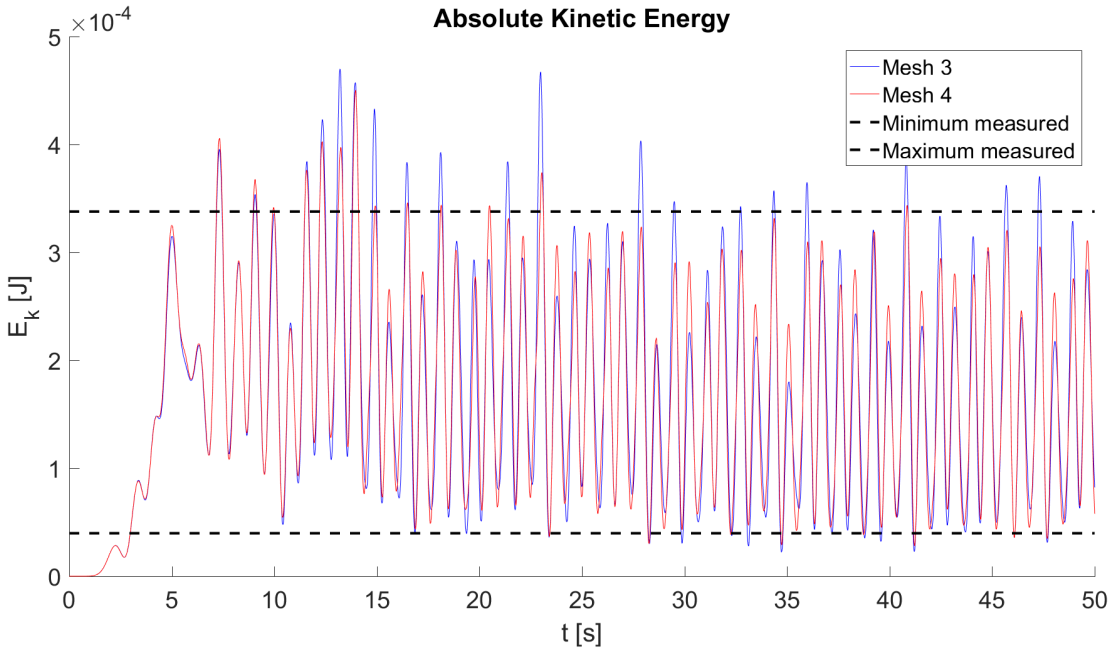

Figure C.23: Kinetic energy content evolution in Area 5 of Mesh 3 and Mesh 4.

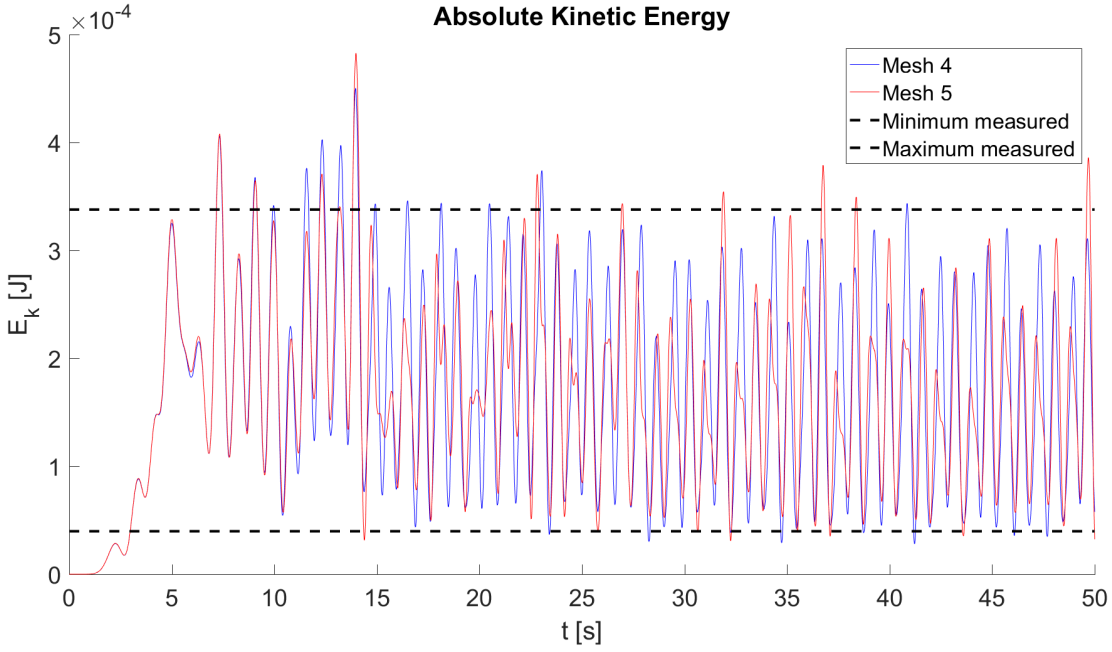

Figure C.24: Kinetic energy content evolution in Area 5 of Mesh 4 and Mesh 5.

Area 6

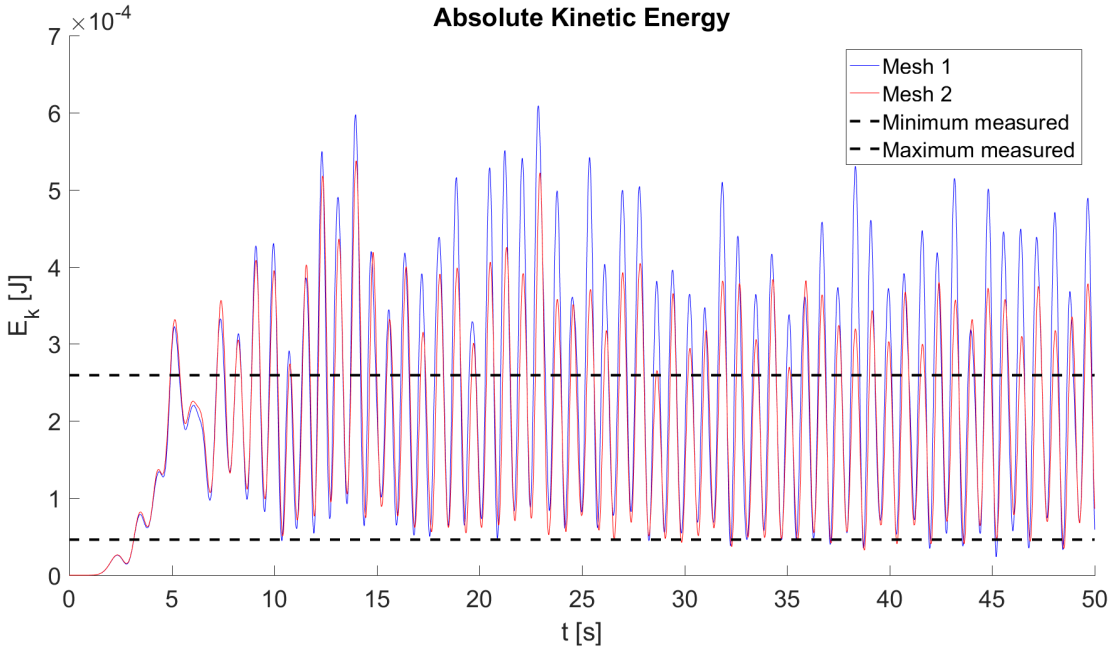

Figure C.25: Kinetic energy content evolution in Area 6 of Mesh 1 and Mesh 2.

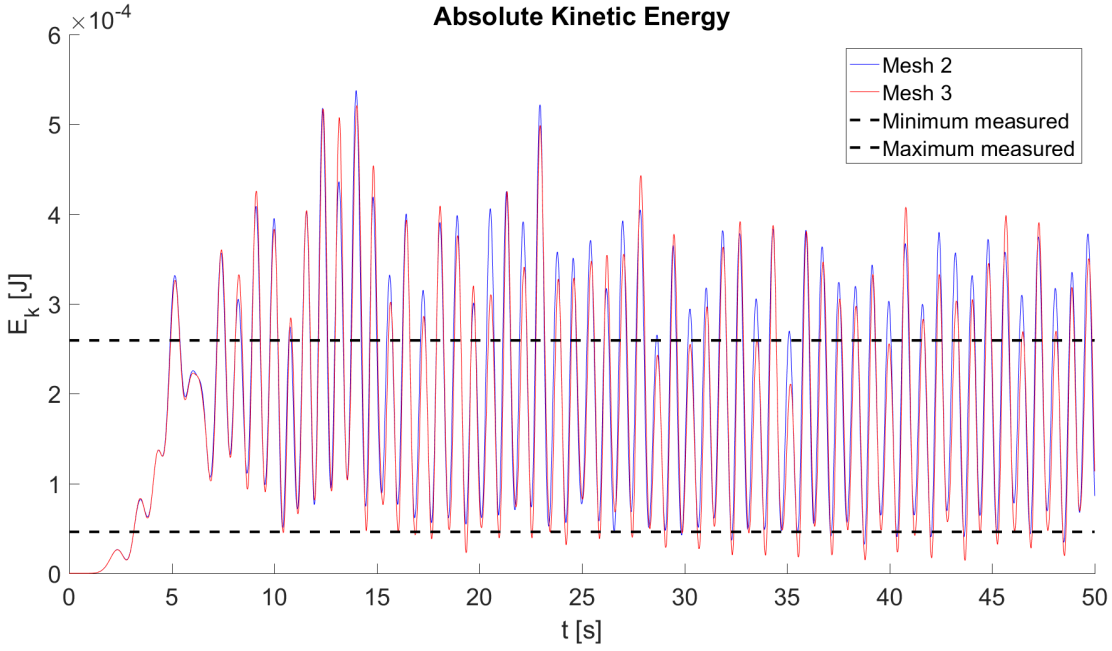

Figure C.26: Kinetic energy content evolution in Area 6 of Mesh 2 and Mesh 3.

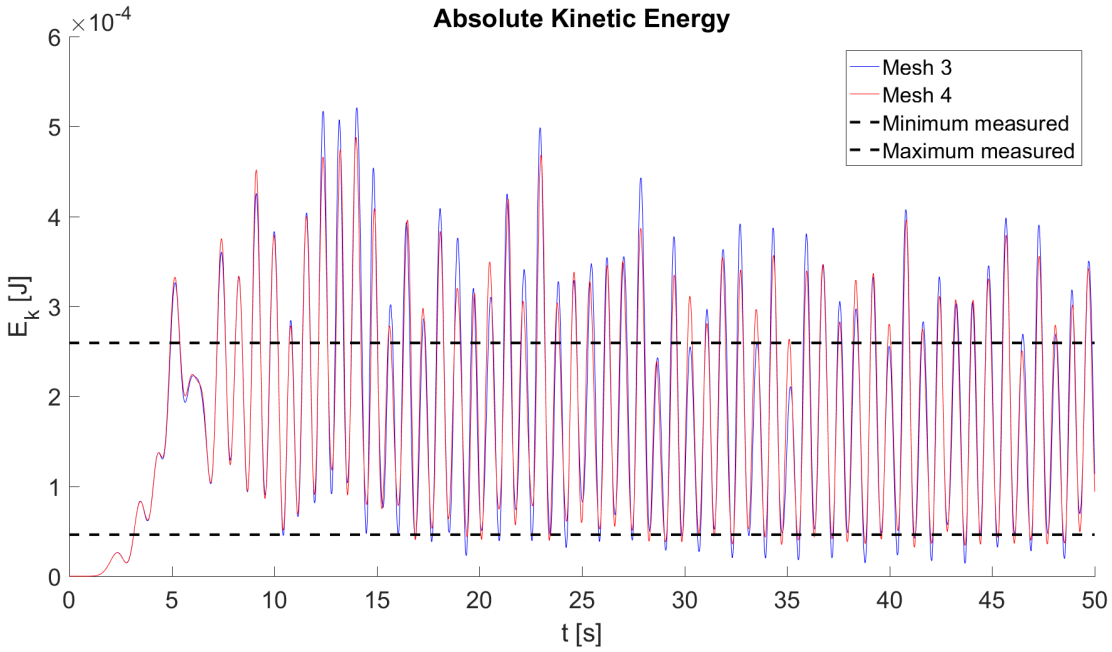

Figure C.27: Kinetic energy content evolution in Area 6 of Mesh 3 and Mesh 4.

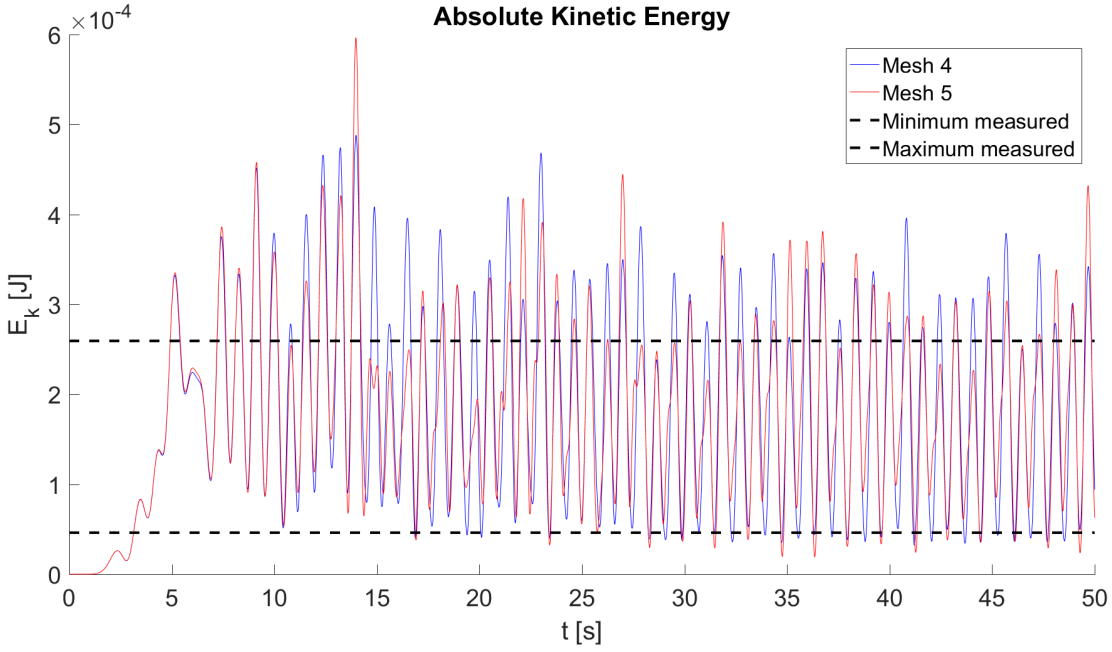

Figure C.28: Kinetic energy content evolution in Area 6 of Mesh 4 and Mesh 5.

Appendix D

Wavelet transform plots

In this appendix, all the studied wavelet transform plots are shown. There are a total of twenty-five Figures. Five Figures for each of the five meshes simulated, corresponding to the evolution of the tank's surface water height in the coordinates: 2.25 m, 3 m, 3.75 m, 4.5 m and 5.25 m.

In each of these Figures, not only the wavelet transform is shown, on top of the wavelet plot the transformed signal is represented in the time domain.

It should be noted that the x-axis of the wavelet transform plot has been standardised to help in its analysis. Instead of time, the x-axis units is the number of times that a wave of the simulated characteristics (as commented in section 4.2, the 'High Wave' profile in Table 4.1) can go from one specific point of the tank to the beach, back to the wave paddle and, finally, arriving at the same specific point again because of reflection.

$x = 2.25\ m$

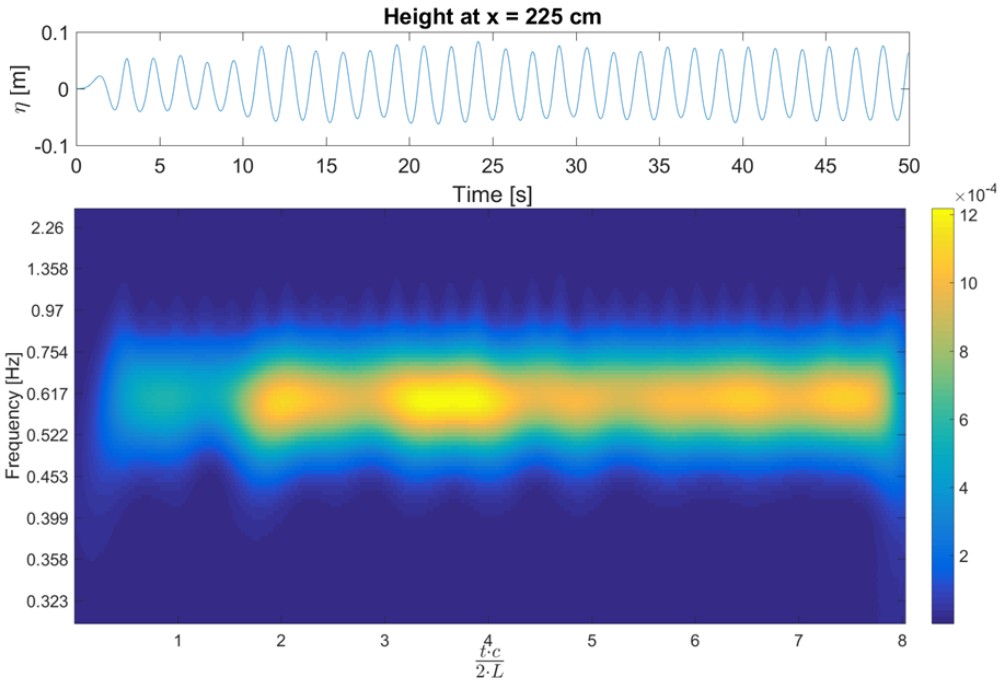

Figure D.1: Wavelet transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 1.

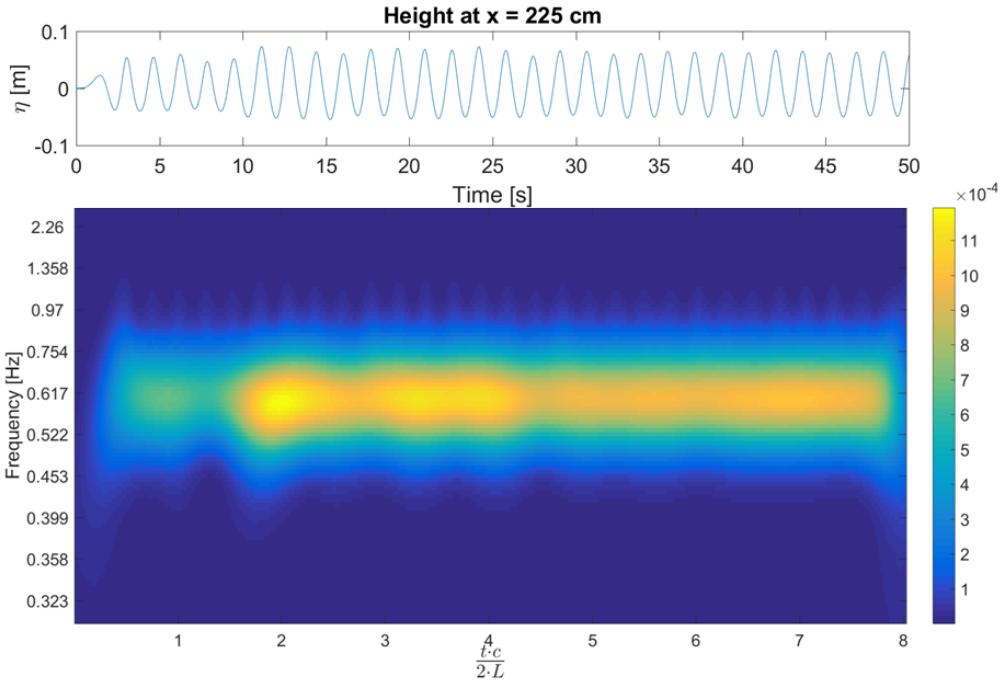

Figure D.2: Wavelet transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 2.

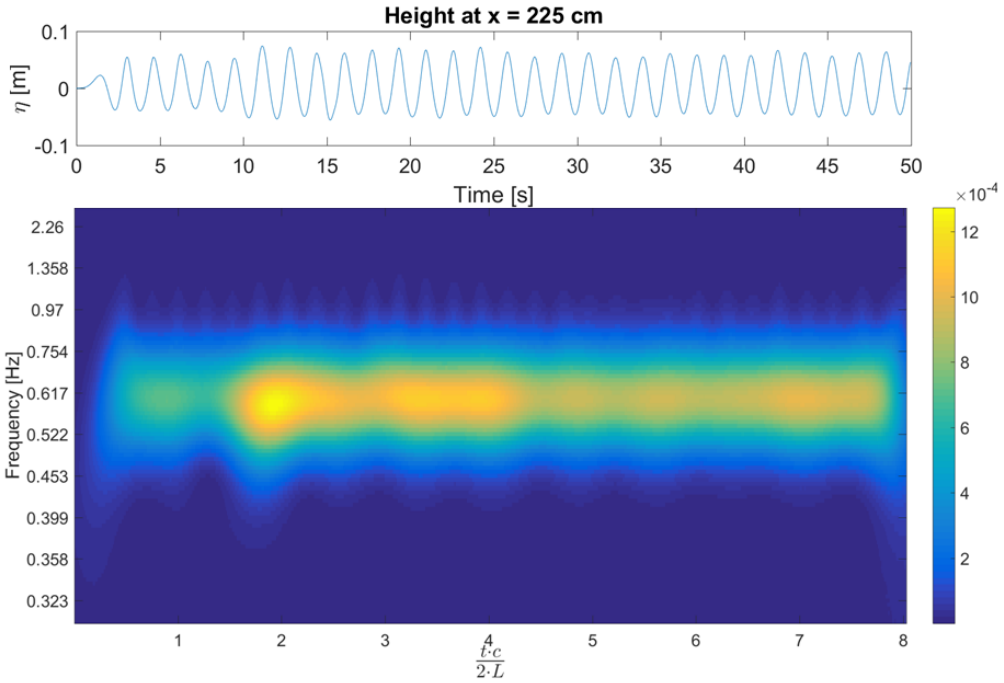

Figure D.3: Wavelet transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 3.

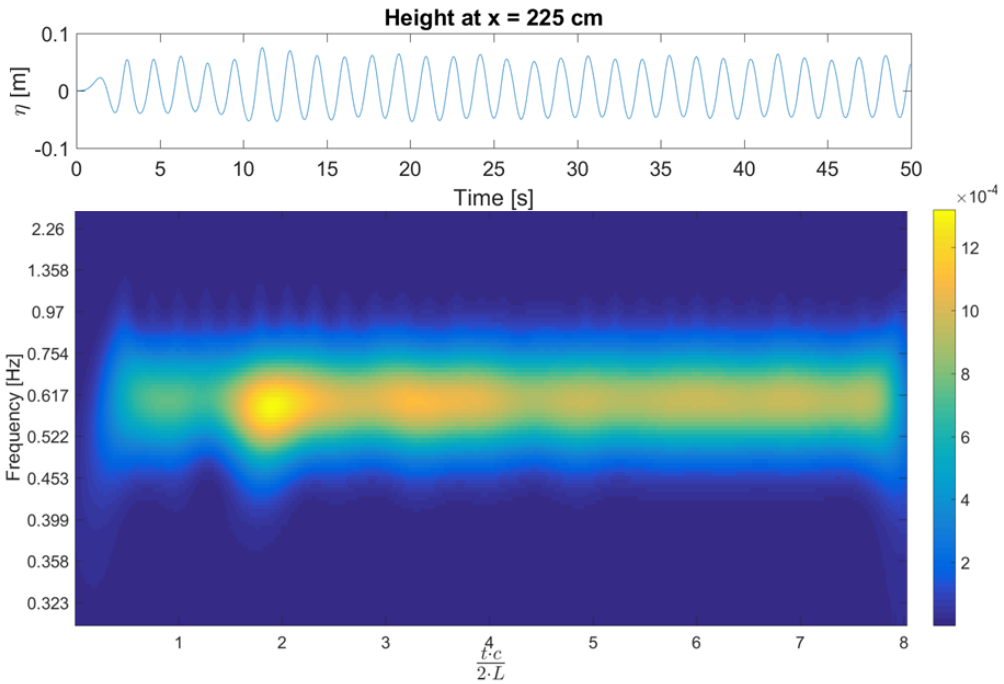

Figure D.4: Wavelet transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 4.

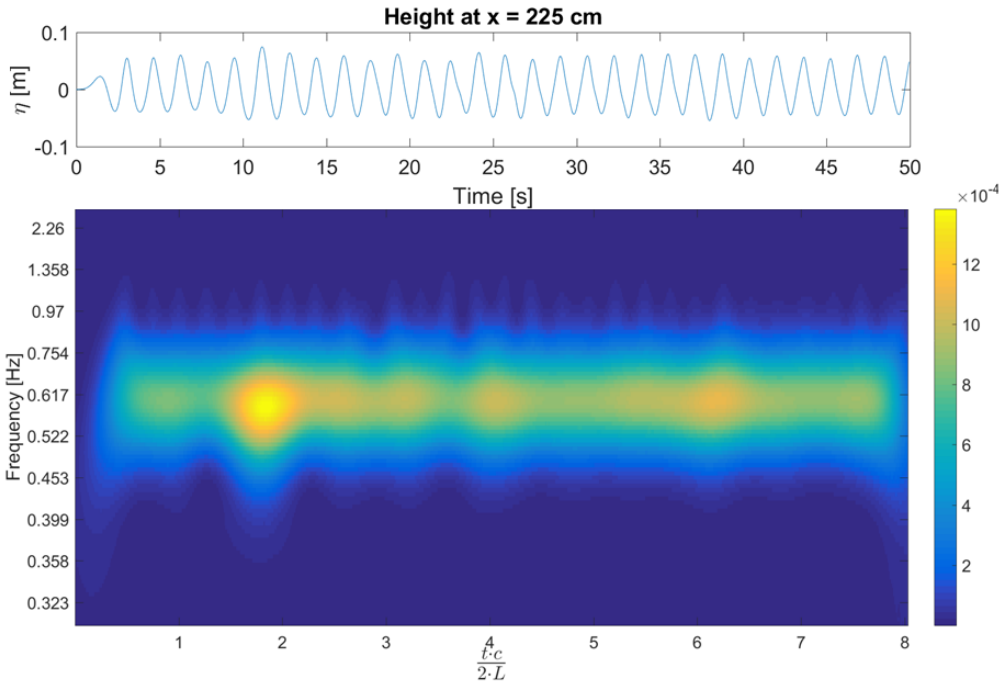

Figure D.5: Wavelet transform of the water height of the water's surface at $x = 2.25\ m$. Mesh 5.

$x = 3.00\ m$

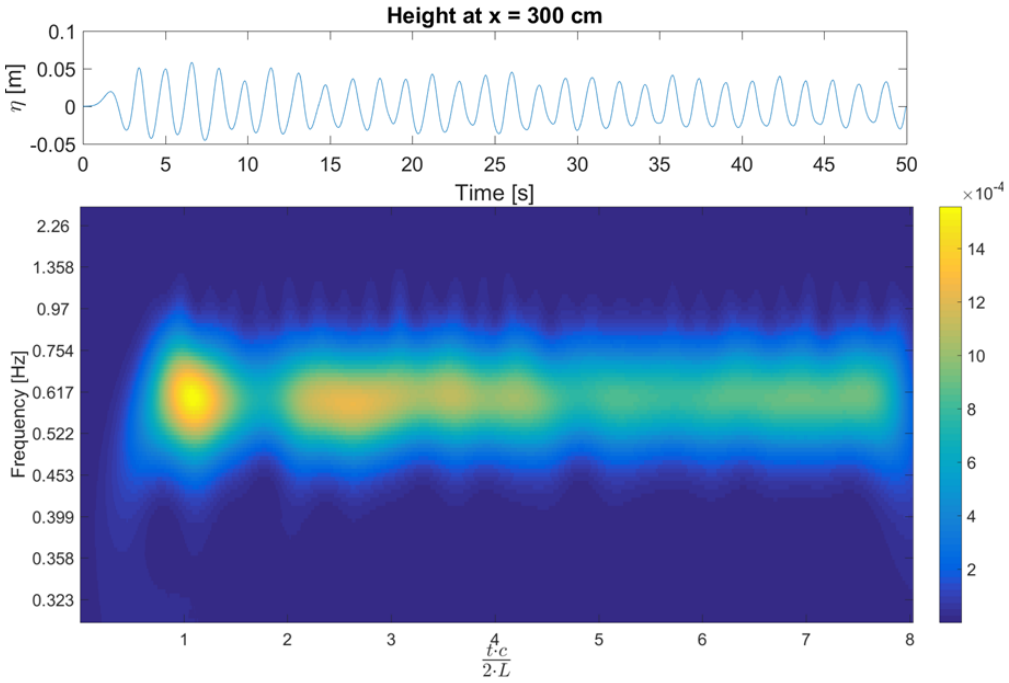

Figure D.6: Wavelet transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 1.

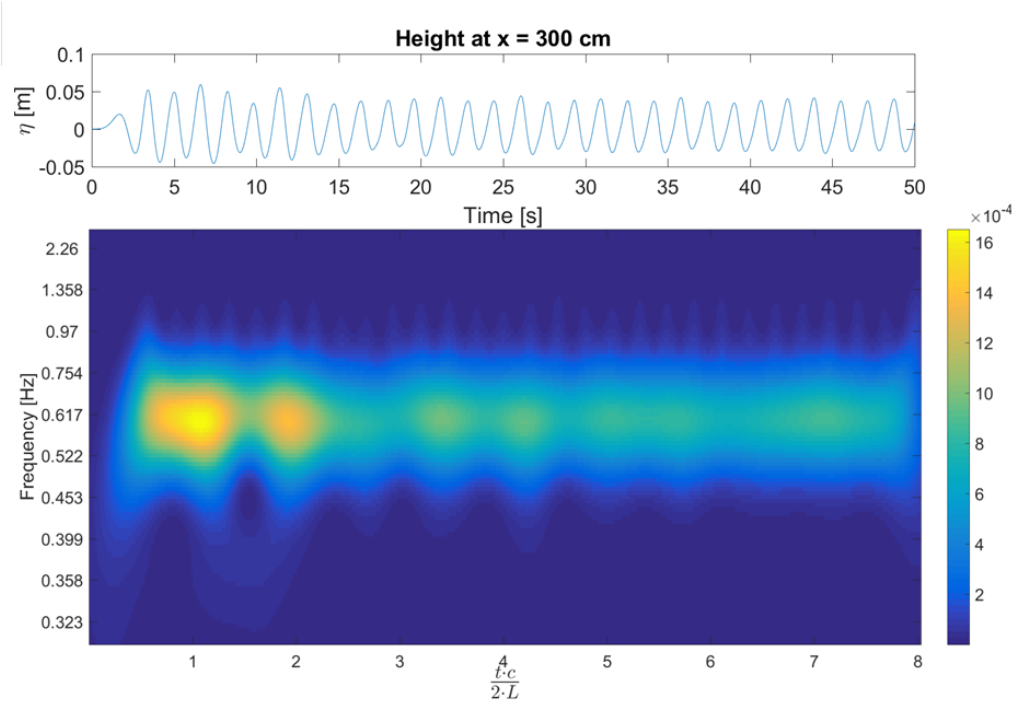

Figure D.7: Wavelet transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 2.

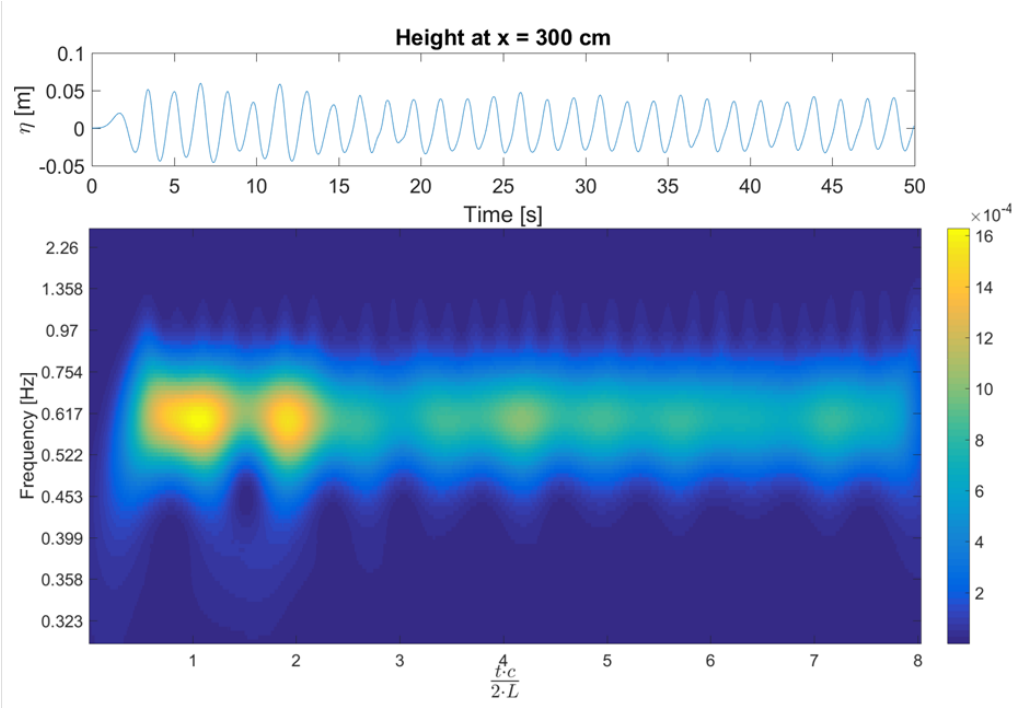

Figure D.8: Wavelet transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 3.

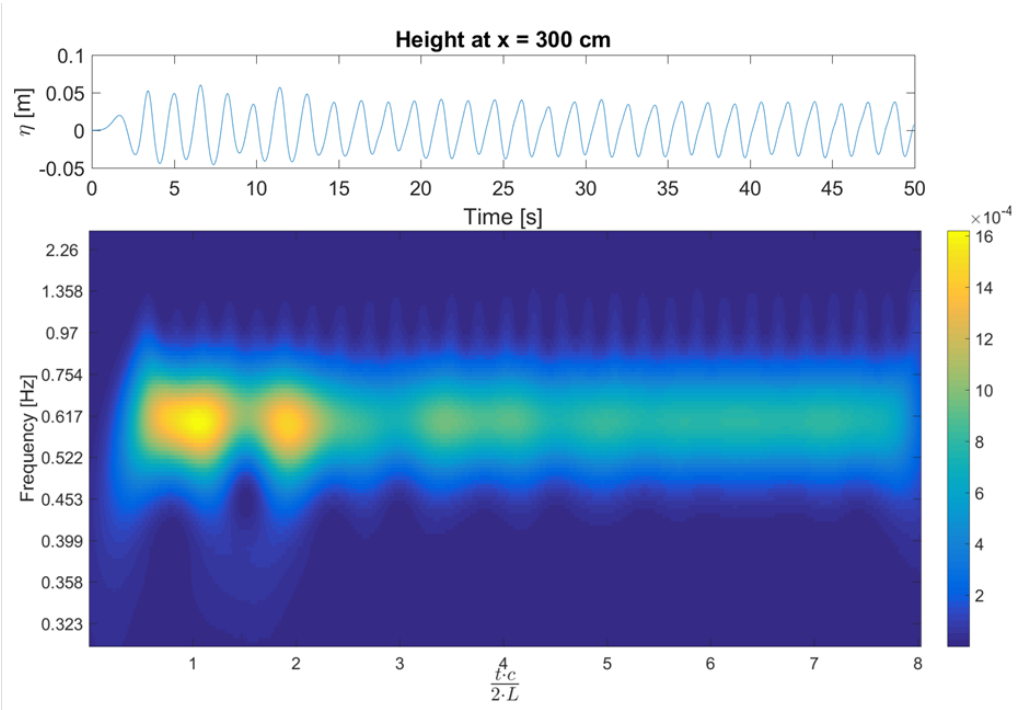

Figure D.9: Wavelet transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 4.

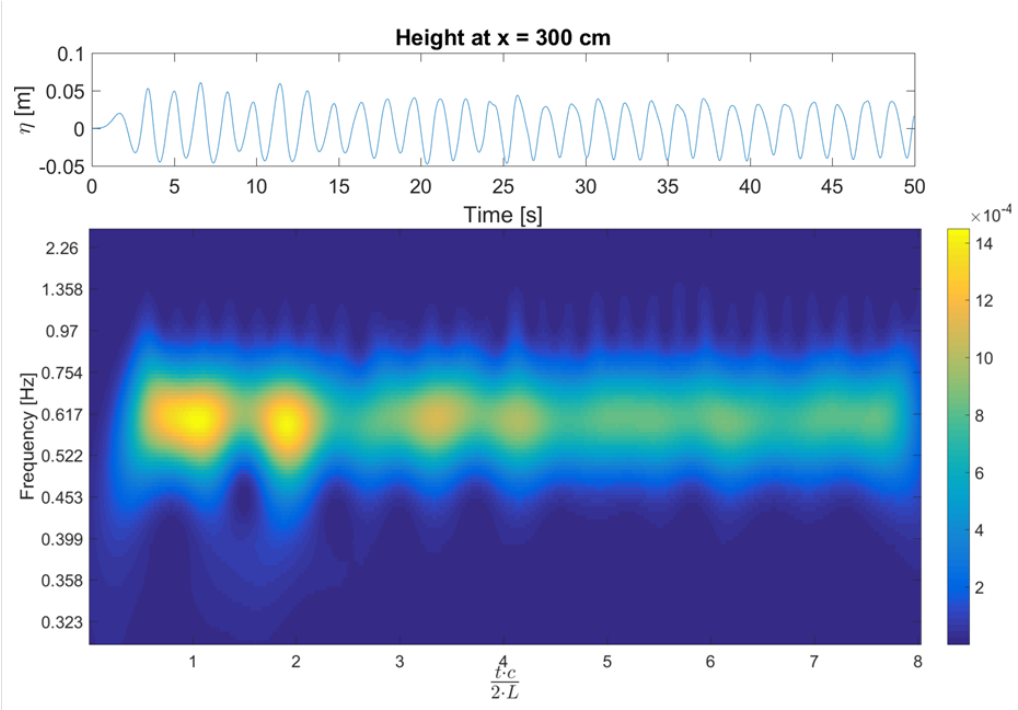

Figure D.10: Wavelet transform of the water height of the water's surface at $x = 3.00\ m$. Mesh 5.

$x = 3.75\ m$

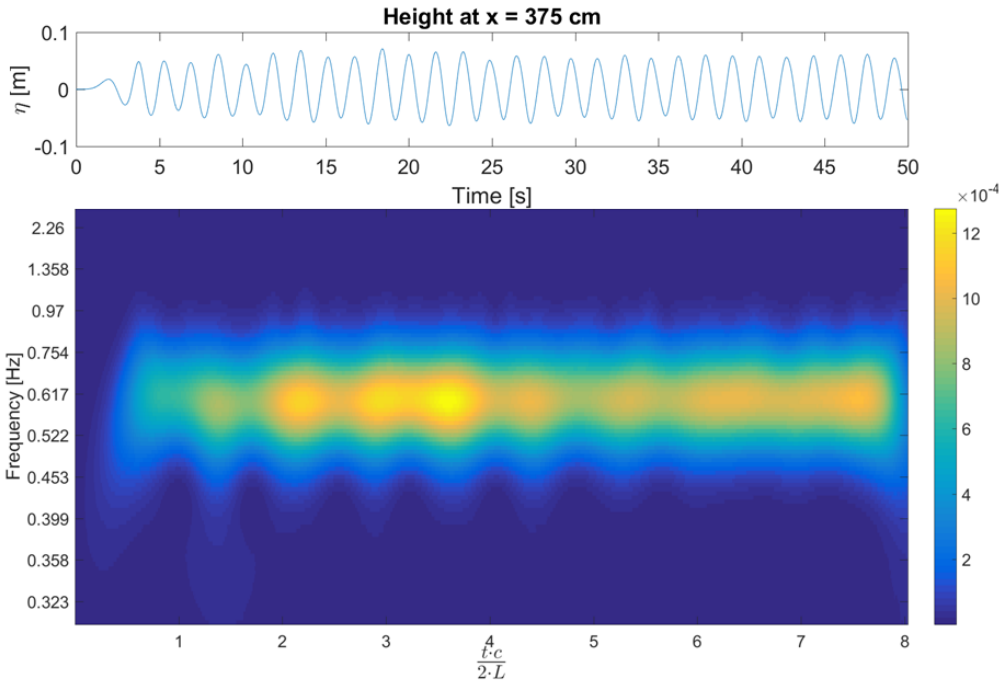

Figure D.11: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 1.

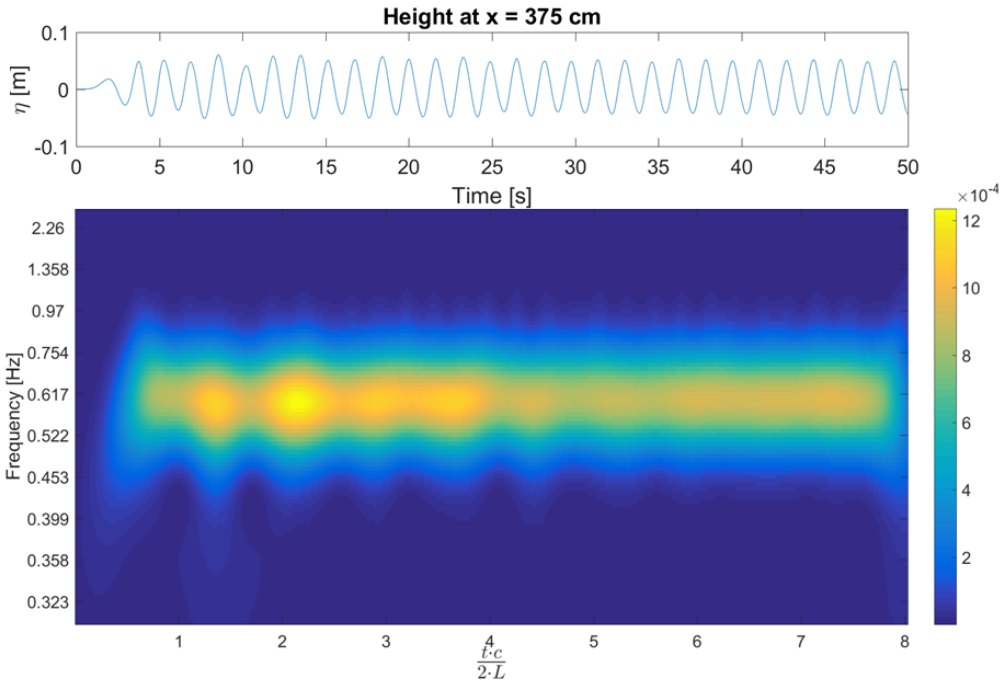

Figure D.12: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 2.

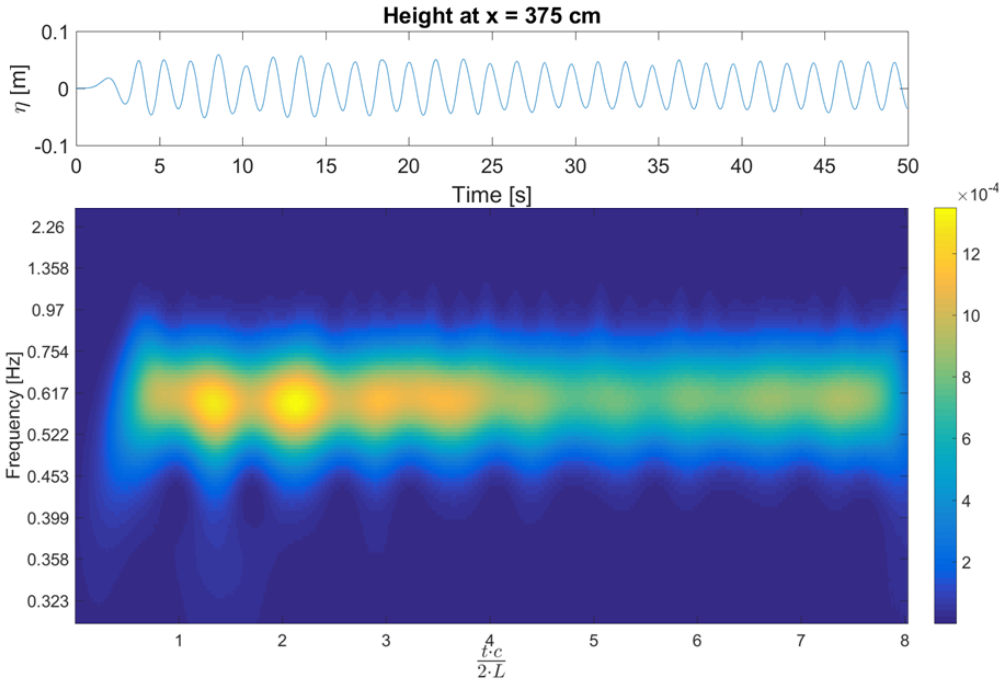

Figure D.13: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 3.

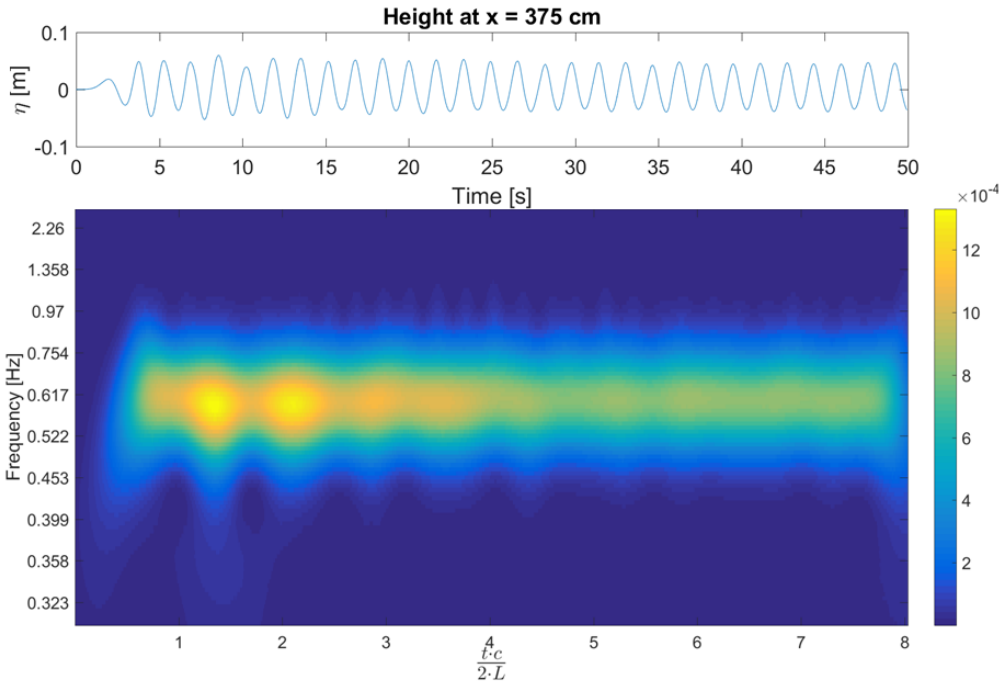

Figure D.14: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 4.

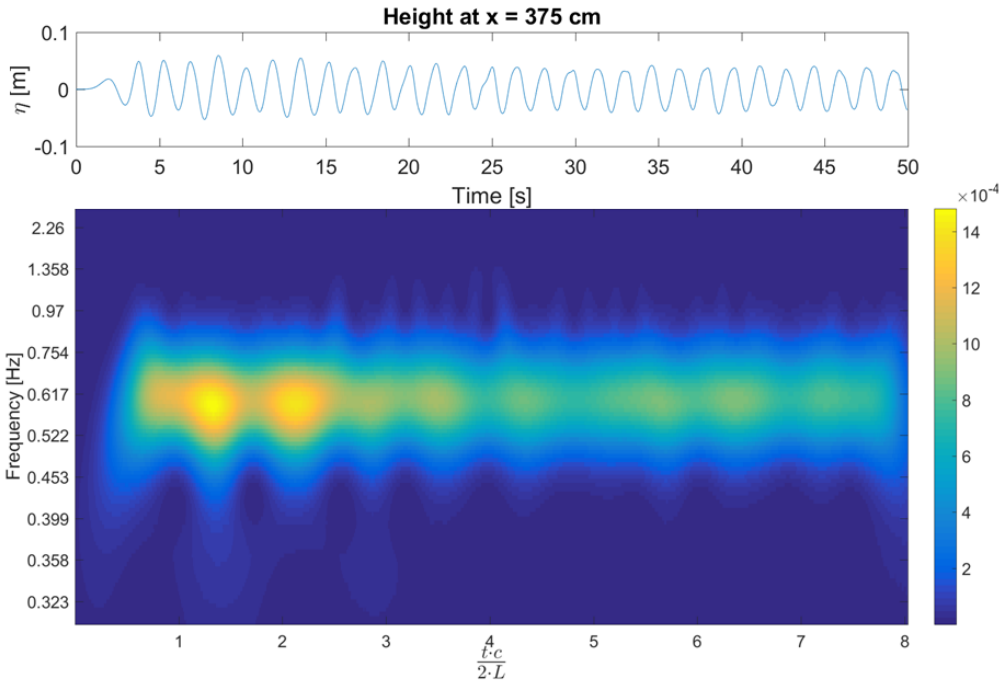

Figure D.15: Wavelet transform of the water height of the water's surface at $x = 3.75\ m$. Mesh 5.

$x = 4.50\ m$

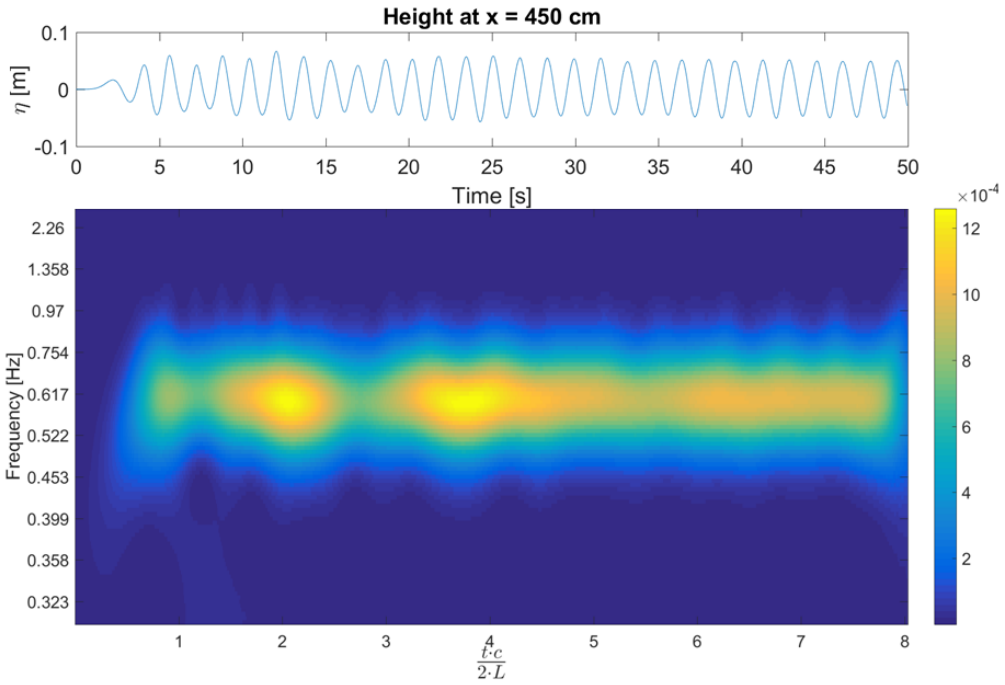

Figure D.16: Wavelet transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 1.

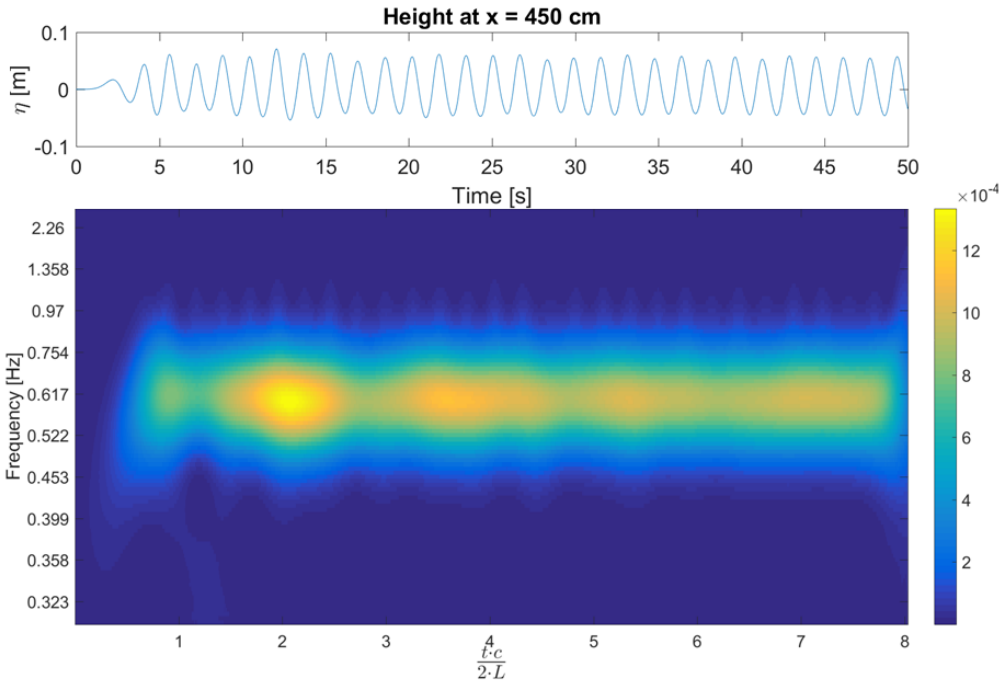

Figure D.17: Wavelet transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 2.

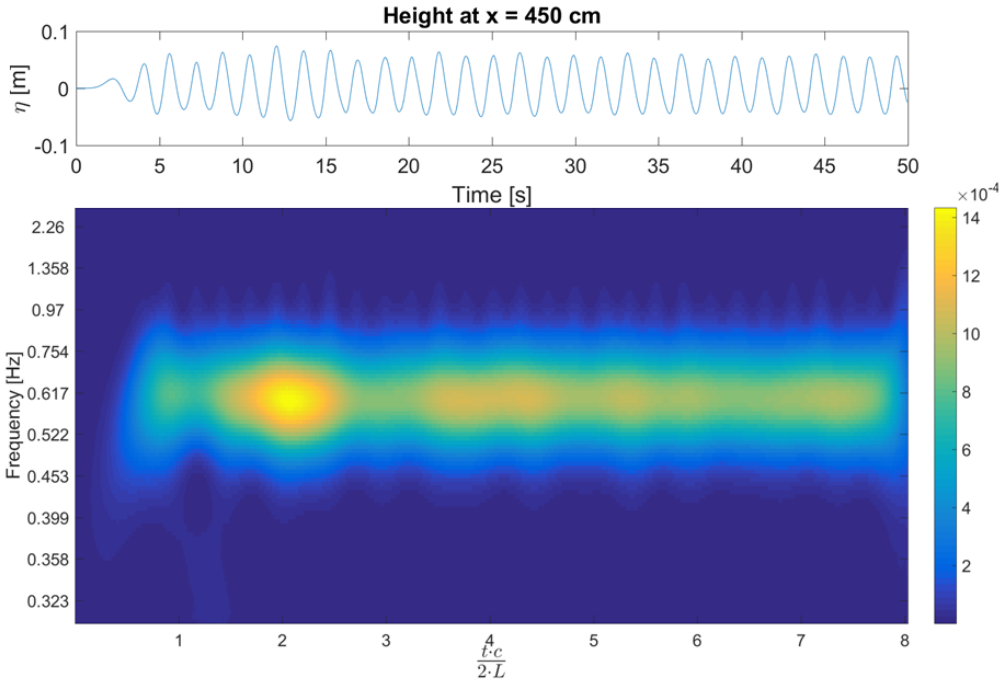

Figure D.18: Wavelet transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 3.

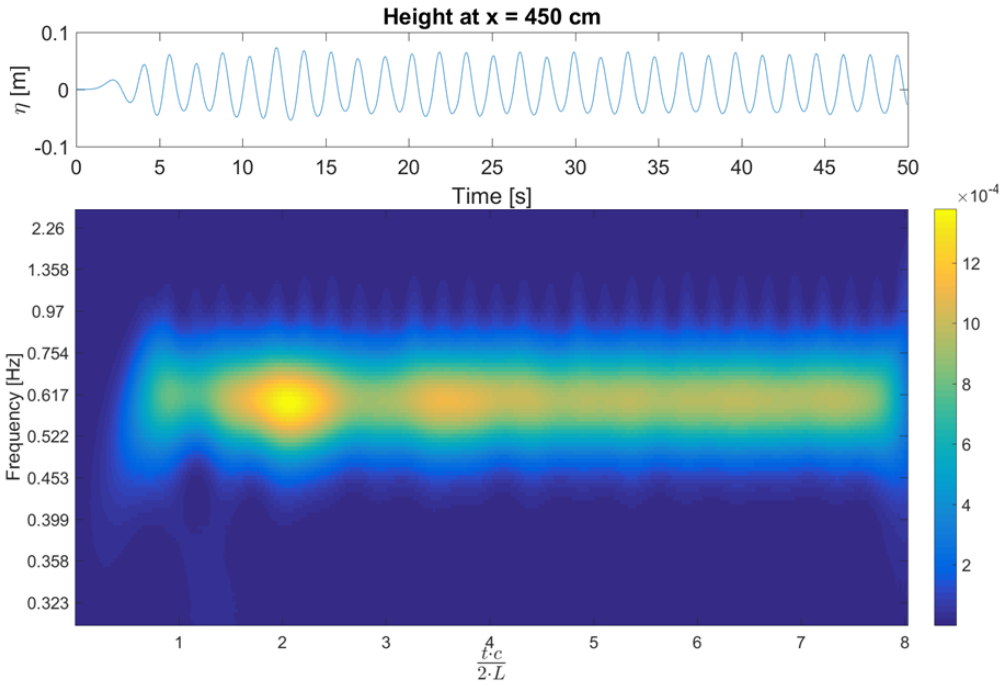

Figure D.19: Wavelet transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 4.

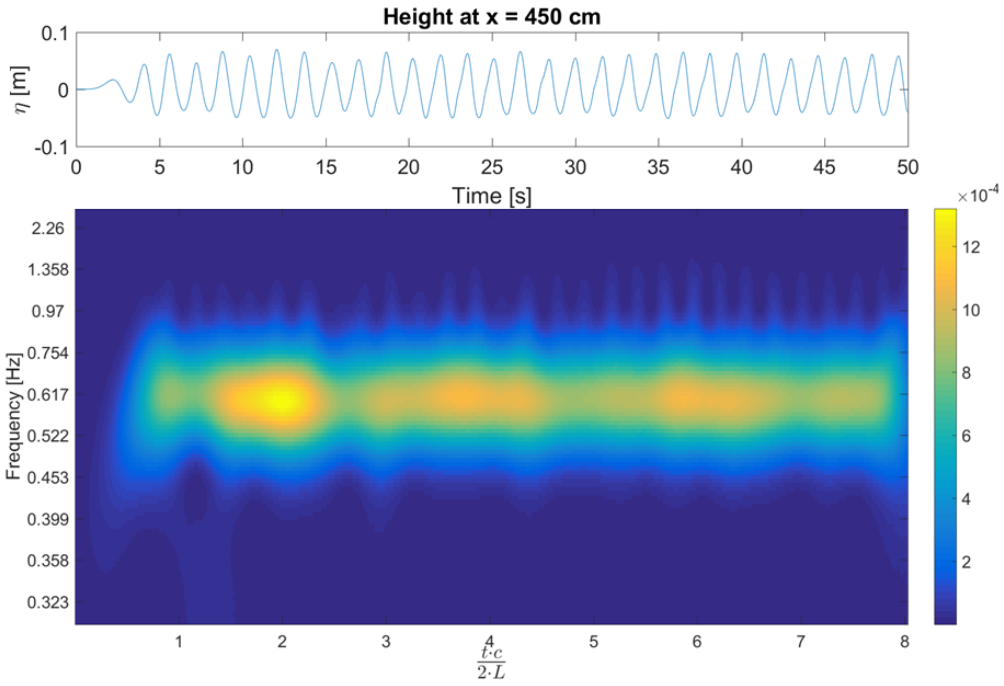

Figure D.20: Wavelet transform of the water height of the water's surface at $x = 4.50\ m$. Mesh 5.

$x = 5.25\ m$

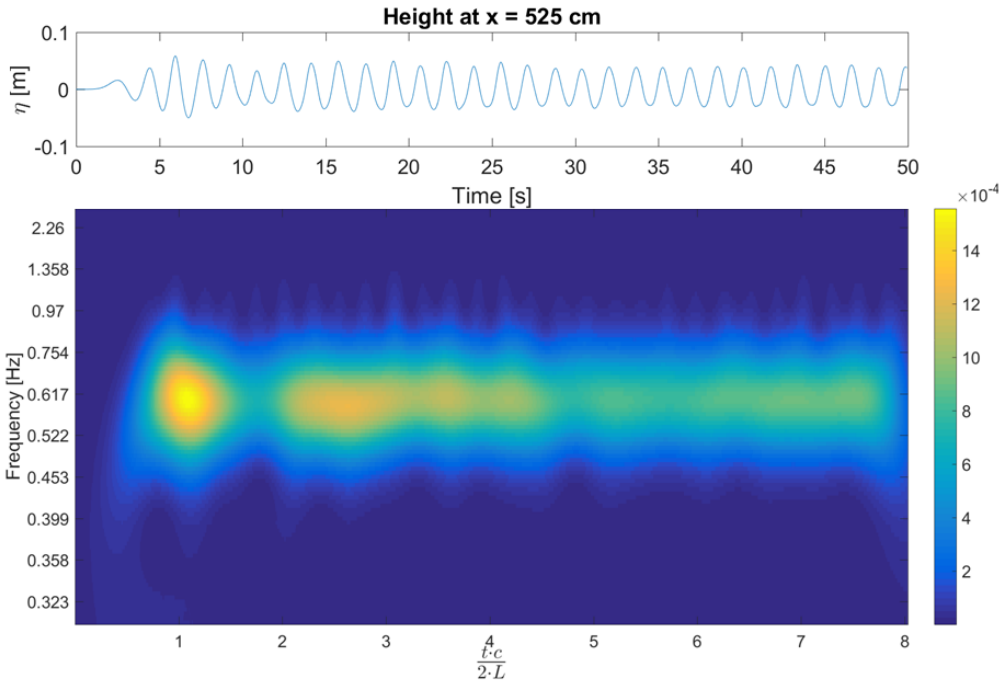

Figure D.21: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 1.

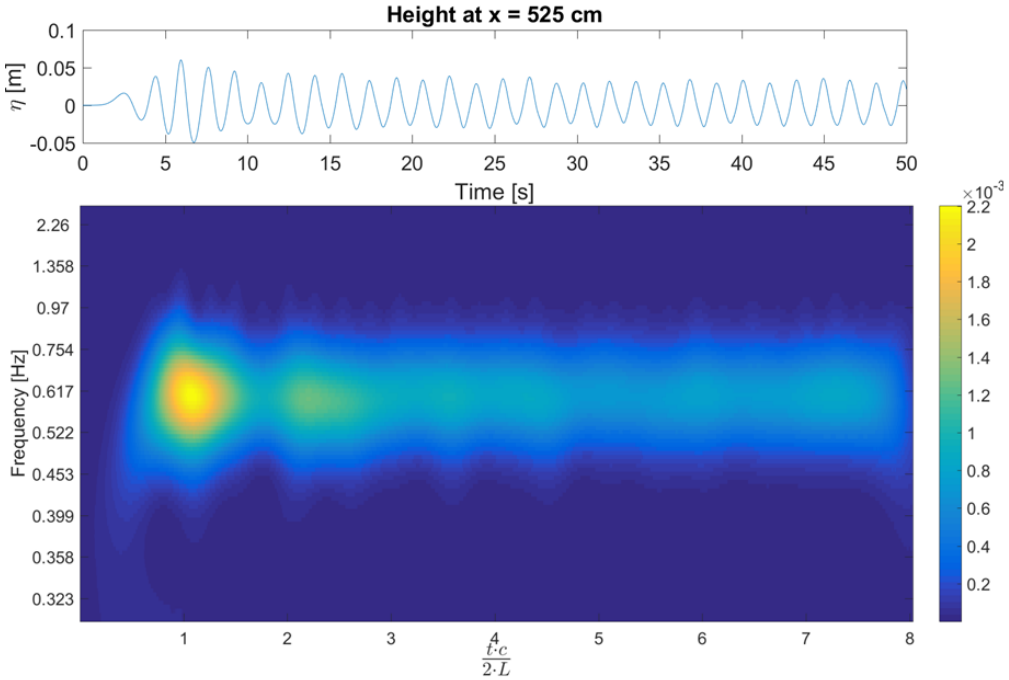

Figure D.22: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 2.

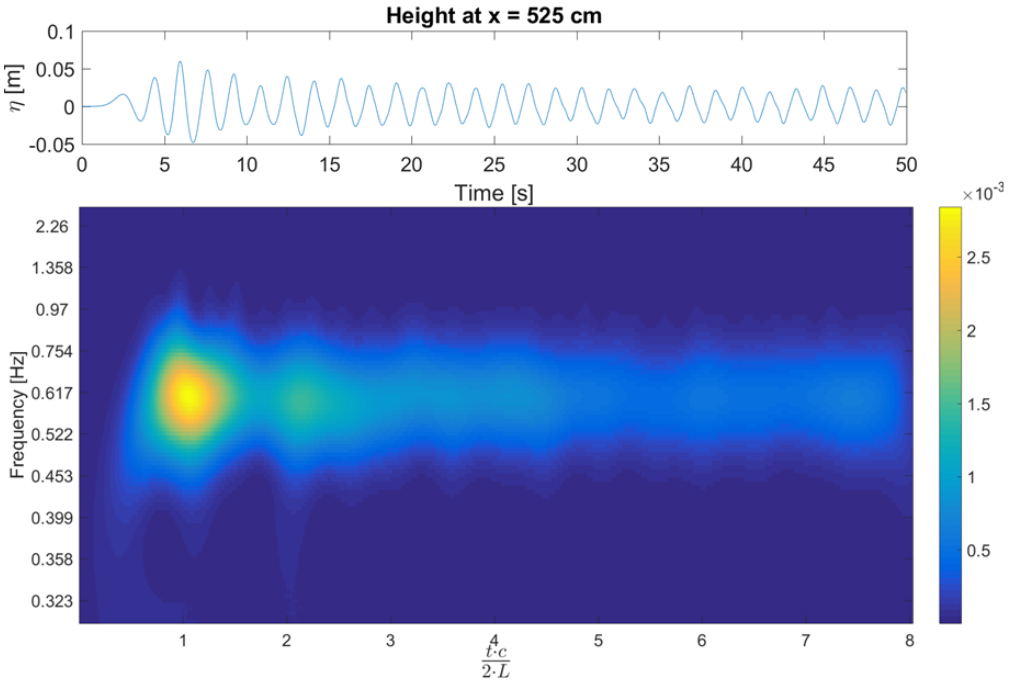

Figure D.23: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 3.

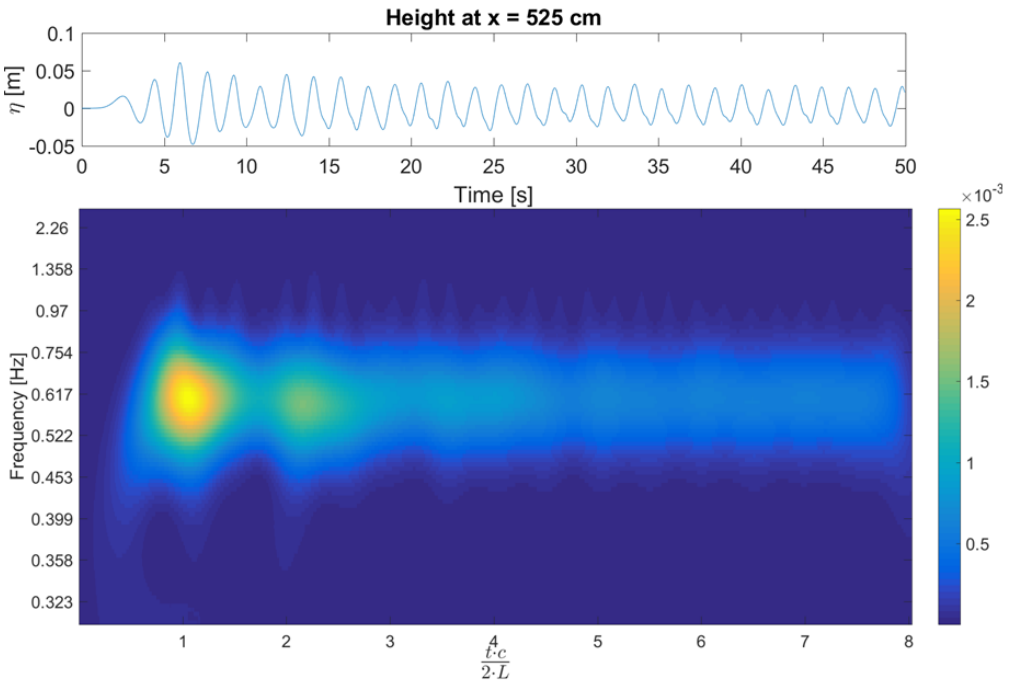

Figure D.24: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 4.

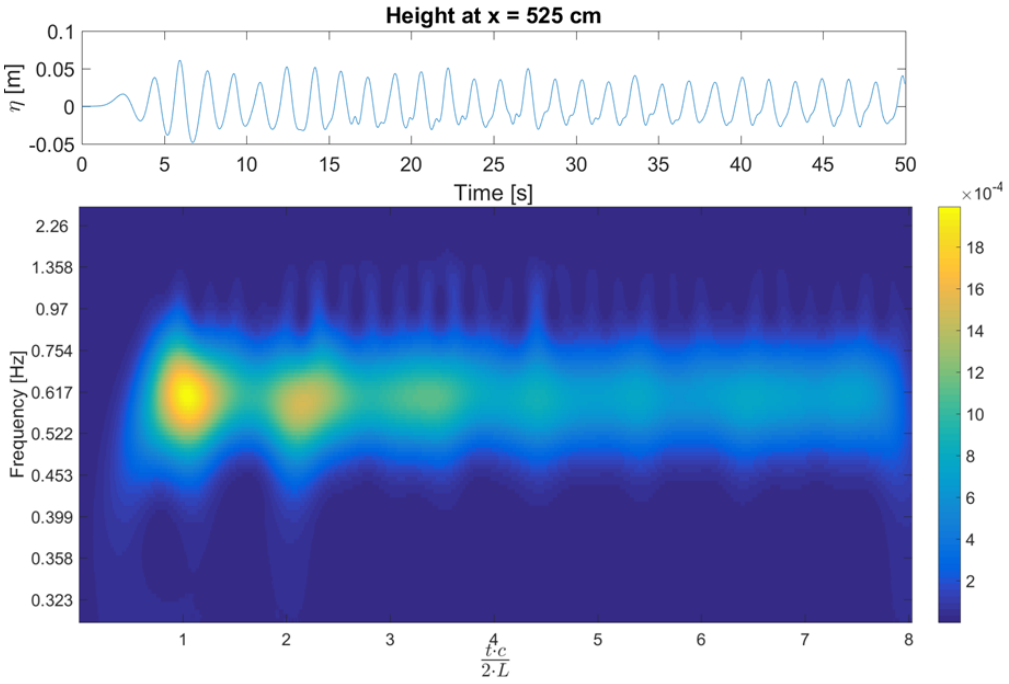

Figure D.25: Wavelet transform of the water height of the water's surface at $x = 5.25\ m$. Mesh 5.